AtAVi

Norme di progetto v1.0.0

Sommario
Questo documento specifica e descrive strumenti, regole e convenzioni utilizzate dal gruppo Co.Code nel corso della realizzazione del *progetto*$_{\text{g}}$ AtAVi.

Versione	1.0.0
Data di redazione	2016-12-19
Redazione	Mattia Bottaro
	Mauro Carlin
Verifica	Luca Bertolini
Approvazione	Simeone Pizzi
Uso	Interno
Distribuzione	prof. Tullio Vardanega
	prof. Riccardo Cardin
	Co.Code

Diario delle modifiche

Versione	Riepilogo	Autore	Ruolo	Data
1.0.8	Stesa sezione relativa alla verifica dei documenti e dei diagrammi UML	chi	ruolo	2017-02-13
1.0.7	Stesa sezione relativa ai processi di configurazione	chi	ruolo	2017-02-13
1.0.6	Spostata la sottosezione "strumenti di versionamento" da "processi organizzativi" a "processi di configurazione"	chi	ruolo	2017-02-13
1.0.5	Stesa struttura e contenuto della parte relativa ai processi di qualità	chi	ruolo	2017-02-13
1.0.4	Aggiunto paragrafo descrittivo della procedura di validazione	chi	ruolo	2017-02-11
1.0.3	Aggiunti strumenti utilizzati nella sezione relativa alla documentazione	chi	ruolo	2017-02-11
1.0.2	Esteso paragrafo riguardante le procedure relative all'analisi dei requisiti.	chi	ruolo	2017-02-11
1.0.1	Completata stesura sezione stile e versionamento codifica. Estesa sezione riguardante il repository	chi	ruolo	2017-02-09
1.0.0	Approvazione documento	Simeone Pizzi	*Responsabile*	2016-12-19
0.3.0	Verifica intero documento	Luca Bertolini	*Verificatore*	2016-12-18
0.2.2	Completata l'intera stesura del documento	Mauro Carlin	*Amministratore*	2016-12-18
0.2.1	Correzione problemi rilevati durante la fase di verifica	Mattia Bottaro	*Amministratore*	2016-12-15
0.2.0	Verifica del documento	Luca Bertolini	*Verificatore*	2016-12-15
0.1.3	Completata stesura della sezione verifica dei processi di supporto	Mauro Carlin	*Amministratore*	2016-12-14
0.1.2	Completata stesura processi organizzativi	Mattia Bottaro	*Amministratore*	2016-12-14

Versione	Riepilogo	Autore	Ruolo	Data
0.1.1	Correzione dei problemi rilevati nella fase di verifica	Mauro Carlin	*Amministratore*	2016-12-13
0.1.0	Verifica del documento	Luca Bertolini	*Verificatore*	2016-12-13
0.0.3	Completata stesura della sezione documentazione dei processi di supporto	Mauro Carlin	*Amministratore*	2016-12-12
0.0.2	Completata stesura processo di sviluppo	Mattia Bottaro	*Amministratore*	2016-12-10
0.0.1	Inizio stesura documento	Mattia Bottaro	*Amministratore*	2016-12-10

Indice

Elenco delle figure

1 Introduzione

1.1 Scopo del documento

Questo documento specifica e definisce le norme da rispettare all'interno del gruppo Co.Code durante lo svolgimento del *progetto*$_g$ AtAVi.
Ogni membro del team è tenuto a visionare il documento e a rispettare le norme in esso contenute. Tali norme permettono di ottenere uniformità nei documenti sviluppati, migliorare l'efficienza del lavoro svolto e ridurre il numero di errori.
In particolare si tratteranno:

- le interazioni tra i membri del team;
- le interazioni del team con componenti esterne;
- le modalità di stesura dei documenti;
- la gestione del *repository*$_g$;
- le modalità di lavoro durante le varie fasi del *progetto*$_g$;
- l'ambiente di lavoro utilizzato.

In caso di modifiche o aggiunte a questo documento è necessario avvisare tutti i membri del gruppo.

1.2 Scopo del *prodotto*$_g$

Si vuole creare un'applicazione web che permetta ad un ospite, in visita all'ufficio di Zero12, di interrogare un assistente virtuale per annunciare la propria presenza, avvisare l'interessato del suo arrivo sul sistema di comunicazione aziendale (*Slack*$_g$) e nel frattempo essere intrattenuto con varie attività.

1.3 Glossario

Allo scopo di evitare ogni ambiguità nel linguaggio e rendere più semplice e chiara la comprensione dei documenti, viene allegato il "*Glossario v1.0.0*". Le parole in esso contenute sono scritte in corsivo e marcate con una 'g' a pedice (p.es. *Parola*$_g$).

1.4 Riferimenti

1.4.1 Riferimenti Normativi

- *Capitolato*$_g$ d'appalto C2 - AtAVi: Accoglienza tramite Assistente Virtuale
  http://www.math.unipd.it/~tullio/IS-1/2016/Progetto/C2.pdf;
- rappresentazione date
  https://en.wikipedia.org/wiki/ISO_8601;
- composizione processo di sviluppo
  https://en.wikipedia.org/wiki/ISO/IEC_12207.

1.4.2 Riferimenti Informativi

- *Git*$_g$
  https://git-scm.com/documentation
- Slide del corso di ingegneria del *software*$_g$
  - http://www.math.unipd.it/~tullio/IS-1/2016/Dispense/L02.pdf
  - http://www.math.unipd.it/~tullio/IS-1/2016/Dispense/L04.pdf
  - http://www.math.unipd.it/~tullio/IS-1/2016/Dispense/L05.pdf
  - http://www.math.unipd.it/~tullio/IS-1/2016/Dispense/L08.pdf
  - http://www.math.unipd.it/~tullio/IS-1/2016/Dispense/L12.pdf

2 Processi primari

2.1 Sviluppo

2.1.1 Scopo

Include le attività e i compiti svolti per creare il *prodotto*$_{\text{g}}$.

2.1.2 Aspettative

Le aspettative della corretta implementazione del processo sono:

- realizzare un *prodotto*$_{\text{g}}$ finale conforme alle richieste del *proponente*$_{\text{g}}$ e che soddisfi le attività di *validazione*$_{\text{g}}$ e *verifica*$_{\text{g}}$;
- fissare gli obiettivi di sviluppo;
- fissare i vincoli tecnologici.

2.1.3 Descrizione

In accordo con lo standard [ISO/IEC 12207], il processo di sviluppo è composto dalle attività di:

- analisi dei requisiti;
- progettazione;
- codifica;
- validazione.

2.1.4 Analisi dei requisiti

2.1.4.1 Scopo dell'attività

Individuare i requisiti del *progetto*$_{\text{g}}$ dalle specifiche del *capitolato*$_{\text{g}}$ e tramite incontri con il proponente. Tale attività produrrà un documento redatto dagli analisti, i quali avranno cura di elencare i *casi d'uso*$_{\text{g}}$ e i requisiti. Tale documento permette di capire le scelte di progettazione effettuate.

2.1.4.2 Aspettative dell'attività

L'attività fissa come scopo la creazione di un documento che elencherà e rappresenterà i requisiti richiesti dal *proponente*$_{\text{g}}$.

2.1.4.3 Descrizione dell'attività

Tutti i requisiti analizzati, utilizzando le specifiche del *capitolato*$_{\text{g}}$ e consultando i proponenti negli incontri effettuati, vanno specificati nell'"*Analisi dei Requisiti v1.0.0*". Per analizzare e trovare i requisiti si utilizza la tecnica dei *casi d'uso*$_{\text{g}}$. Il tracciamento dei requisiti avviene tramite l'applicativo PragmaDB.

2.1.4.4 Studio di fattibilità

Il *Responsabile* di *progetto*$_g$ deve organizzare delle riunioni preventive, per permettere lo scambio di opinioni tra i membri del gruppo sui capitolati proposti. Il documento *prodotto*$_g$ da queste riunioni è lo "*Studio di Fattibilità v1.0.0*", il quale viene realizzato dagli *Analisti*. Essi devono descrivere i seguenti punti:

- **Dominio tecnologico e applicativo**: si dà una valutazione prendendo in considerazione la conoscenza attuale delle tecnologie richieste dal *capitolato*$_g$ in analisi da parte dei membri del gruppo;
- **Interesse strategico**: si valuta l'interesse strategico del gruppo di *progetto*$_g$ in relazione al *capitolato*$_g$ in analisi;
- **Individuazione dei rischi**: si analizzano i possibili rischi in cui si può incorrere nel *capitolato*$_g$ in analisi.

2.1.4.5 *Casi d'uso*$_g$

Ogni caso d'uso è così composto:

- **Codice identificativo**: codice univoco del caso d'uso in esame;
- **Titolo**: indica il titolo del caso d'uso;
- **Diagramma *UML*$_g$**: rappresenta graficamente il caso d'uso;
- **Attori primari**: indica gli attori primari coinvolti;
- **Descrizione**: chiara, precisa e concisa descrizione del caso d'uso;
- **Precondizione**: indica la situazione che deve essere vera prima dell'esecuzione del caso d'uso;
- **Postcondizione** indica la situazione che deve essere vera dopo l'esecuzione del caso d'uso;
- **Scenario principale**: descrizione composta dal flusso dei *casi d'uso*$_g$ figli;
- **Scenari alternativi**: descrizione composta dai *casi d'uso*$_g$ che non appartengono al flusso principale di esecuzione.

2.1.4.6 Codice identificativo dei *casi d'uso*$_g$

Ogni caso d'uso ha un proprio codice identificativo che rispetta il seguente formalismo:

UC{Codice}

dove:

- **Codice**: indica il codice identificativo del requisito, è univoco e deve essere identificato in forma gerarchica.

2.1.4.7 Requisiti

Ogni requisito è così composto:

- **Codice identificativo**: codice univoco del requisito;
- **Descrizione**: una breve descrizione, deve essere meno ambigua possibile;
- **Fonti**: identifica la fonte dalla quale è stato identificato il requisito.

2.1.4.8 Codice identificativo dei requisiti

Ogni requisito individuato avrà un codice identificativo univoco così formato:

R{Tipo}{Importanza}{Codice}

dove:

- **Tipo**: può assumere uno di questi valori:
  - **F**: indica un requisito funzionale;
  - **Q**: indica un requisito di qualità;
  - **P**: indica un requisito prestazionale;
  - **V**: indica un requisito di vincolo.
- **Importanza**: può assumere uno di questi valori:
  - **O**: indica un requisito obbligatorio;
  - **D**: indica un requisito desiderabile;
  - **F**: indica un requisito facoltativo.
- **Codice**: indica il codice identificativo del requisito, è univoco e deve essere identificato in forma gerarchica.

2.1.4.9 *UML*$_{\text{g}}$

Viene utilizzata la versione corrente alla stesura del documento, ovvero la 2.5.

2.1.5 Progettazione

2.1.5.1 Scopo dell'attività

L'attività di progettazione definisce le linee essenziali della struttura del *prodotto*$_{\text{g}}$ *software*$_{\text{g}}$ in funzione dei requisiti individuati dall'analisi. L'obiettivo del processo consiste nella stesura dei documenti: *"Specifica Tecnica"* e *"Definizione di Prodotto"*.
Compresi pienamente quali siano i requisiti del problema e approfondendo la progettazione a moduli abbastanza semplici da essere capiti da una sola persona, si otterranno le istruzioni necessarie ai *Programmatori* per sviluppare il prodotto.

2.1.5.2 Aspettative dell'attività

Il processo porta alla formazione dei documenti sopracitati, i quali garantiscono affidabilità e coerenza.

2.1.5.3 Descrizione dell'attività

La progettazione deve rispettare tutti i vincoli e i requisiti concordati tra i componenti del gruppo e i proponenti. I documenti derivati da questa attività sono:

- **Specifica tecnica**: descrive la progettazione ad alto livello relativa all'architettura dell'applicazione e dei singoli componenti. Il documento specifica i diagrammi *UML*$_{\text{g}}$ ed i design pattern utilizzati per realizzare l'architettura definendo inoltre i test necessari alla *verifica*$_{\text{g}}$;

- **Definizione di *prodotto*$_{\mathbf{g}}$**: descrive in dettaglio la progettazione di *sistema*$_{\mathrm{g}}$, integrando quanto scritto nella Specifica Tecnica. Il documento specifica i diagrammi *UML*$_{\mathrm{g}}$ e le definizioni delle classi definendo inoltre i test necessari alla *verifica*$_{\mathrm{g}}$.

2.1.5.4 Specifica tecnica

- **Diagrammi *UML*$_{\mathbf{g}}$**:
  - diagrammi delle classi;
  - diagrammi dei *package*$_{\mathrm{g}}$;
  - diagrammi di attività;
  - diagrammi di sequenza.
- ***Design pattern*$_{\mathbf{g}}$**: devono essere descritti i *design pattern*$_{\mathrm{g}}$ utilizzati per realizzare l'architettura. Ogni design pattern deve essere accompagnato da una descrizione ed un diagramma, che ne esponga il significato e la struttura;
- **Tracciamento delle componenti**:
- **Test di integrazione**: devono essere definite delle classi di *verifica*$_{\mathrm{g}}$, utili a verificare che ogni componente del *sistema*$_{\mathrm{g}}$ funzioni nella maniera appropriata.

2.1.5.5 Definizione di *prodotto*$_{\mathbf{g}}$

- **Diagrammi *UML*$_{\mathbf{g}}$**:
  - diagrammi delle classi;
  - diagrammi di attività;
  - diagrammi di sequenza.
- **Definizioni delle classi**: ogni classe progettata deve essere descritta in modo da spiegarne lo scopo e definirne le funzionalità ad essa associate.
- **Tracciamento delle classi**: ogni requisito deve essere tracciato, in modo da poter risalire alle classi ad esso associate.
- **Test di unità**: devono essere definiti dei test di unità utili a verificare che le componenti del *sistema*$_{\mathrm{g}}$ funzionino nel modo previsto.

2.1.6 Codifica

2.1.6.1 Scopo dell'attività

Lo scopo dell'attività è l'implementazione del *prodotto*$_{\mathrm{g}}$, concretizzando la soluzione tramite la codifica.

2.1.6.2 Aspettative dell'attività

L'aspettativa dell'attività è un *prodotto*$_{\mathrm{g}}$ corretto, ovvero stabile, affidabile, funzionale e che soddisfi i requisiti.

2.1.6.3 Descrizione dell'attività

L'attività deve rispettare i compiti e gli strumenti espressi nel "*Piano di Progetto v2.0.0*".

2.1.6.4 Stile

Al fine di rendere il codice più leggibile possibile, il gruppo dovrà rispettare le seguenti regole:

- all'inizio di ogni file dovrà essere presente un'intestazione contenete il nome dell'autore, la data di creazione, l'utilità del file e il diario delle modifiche;
- ogni modifica apportata dovrà essere descritta nell'intestazione;
- scrivere dei commenti in italiano qualora certe porzioni di codice possano risultare poco chiare o complesse;
- evitare di commentare porzioni di codice dal funzionamento banale;
- il codice deve essere identato al meglio, in particolare se uno o più statement sono interni a una porzione di codice, essi dovranno rientrare di quattro spazi, come nell'esempio che segue:

```
if(condition)
{
    a=b;
    b=c;
}
```

- le parentesi graffe che vanno a delimitare un blocco di codice dovranno trovarsi al di sotto della sua segnatura, come negli esempi che seguono:

```
while(condition)
{
    operazioni
}

void foo()
{
    operazioni
}
```

- evitare più di una istruzione per riga;
- le costanti devono essere scritte in maiuscolo;
- i nomi delle variabili devono rispettare le seguenti regole:
  - avere un nome più esplicativo possibile;
  - se il nome è composto da più parole, esse devono essere divise da un carattere di underscore ' _ ';
  - essere, se possibile, inizializzate ad un valore.
- i nomi dei metodi e funzioni devono rispettare le seguenti regole:
  - avere un nome più esplicativo possibile;
  - se composto da più parole, il nome deve seguire la notazione camelCase;
  - essere associati ad un contratto, il quale ne spiega l'utilità;
  - evitare un numero eccessivo di parametri.
- evitare l'eccessivo innestarsi di statement;
- linee troppo lunghe dovranno essere spezzate in più parti.

Il codice che segue vuole essere un esempio delle regole appena descritte.

```
/*
Autore : Mario Rossi
Data : 2017-04-12
Questo file contiene la definizione e implementazione della classe Automobile
Modifiche:
Luigi Verdi, 2017-04-13, aggiunto metodo getNumeroDiPorte, versione 0.0.2
Mario Rossi, 2017-04-12, creato il file, versione 0.0.1
*/
public class Automobile
{
    private int numero_di_porte;
    private double final PREZZO_BASE=5000;
    /*
    Il metodo seguente ritorna il numero di ruote dell'automobile.
    In caso il numero di ques'ultime sia negativo, si segnala l'errore ritornando 0.
    */
    public getNumeroDiPorte()
    {
        if(numero_di_porte < 0)
            return 0; // si è verificato un errore
        return numero_di_porte;
    }
}
```

Figura 1: Esempio di stile di codifica

2.1.6.5 Versionamento

La versione del codice viene inserita all'interno dell'intestazione del file e segue il seguente formalismo:

x.y.z

dove:

- x: è l'indice di versione principale, un incremento di tale indice rappresenta un avanzamento della versione stabile, di conseguenza i valori degli indici y e z devono essere azzerati;
- y: è l'indice di verifica del file, un incremento di tale indice comporta l'azzeramento dell'indice z;
- z: è l'indice di modifica parziale, un incremento di tale indice rappresenta una modifica rilevante, come per esempio la rimozione o l'aggiunta di una istruzione. La versione 1.0.0 deve rappresentare la prima versione del file completo e stabile, cioè quando le sue fun- zionalità obbligatorie sono state definite e si considerano funzionanti. Solo dalla versione 1.0.0 è possibile testare il file, con degli appositi test definiti, per verificarne l'effettivo funzionamento.

2.1.6.6 Ricorsione

La *ricorsione*$_g$ va evitata. Se non risulta accettabile convertirla in *iterazione*$_g$, bisogna fornirne la prova di terminazione e l'analisi del costo in termini di spazio.

2.1.6.7 Test

DA DEFINIRE

2.1.7 Procedure

2.1.7.1 Inserimento di un attore in PragmaDB

Per inserire un attore in PragmaDB è necessario applicare la seguente procedura:

- effettuare l'accesso in PragmaDB;
- selezionare la voce "Attori";
- selezionare la voce "Inserisci Attore";

- inserire nome e descrizione dell'attore.

2.1.7.2 Inserimento di un caso d'uso in PragmaDB

Per inserire un caso d'uso in PragmaDB è necessario applicare la seguente procedura:

- effettuare l'accesso in PragmaDB;
- selezionare la voce "Use Case";
- selezionare la voce "Inserisci Use Case";
- popolare tutti i campi richiesti;
- inserire gli attori coinvolti;
- inserire i requisiti correlati.

2.1.7.3 Inserimento di una fonte in PragmaDB

Per inserire una fonte in PragmaDB è necessario applicare la seguente procedura:

- effettuare l'accesso in PragmaDB;
- selezionare la voce "Fonti";
- selezionare la voce "Inserisci Fonte";
- inserire nome e descrizione della fonte.

2.1.7.4 Inserimento di un requisito in PragmaDB

Per inserire un requisito in PragmaDB è necessario applicare la seguente procedura:

- effettuare l'accesso in PragmaDB;
- selezionare la voce "Requisiti";
- selezionare la voce "Inserisci Requisito";
- popolare tutti i campi richiesti;
- inserire la fonte correlata;
- inserire l'use case correlato.

2.1.7.5 Inserimento di un componente in PragmaDB

2.1.7.6 Inserimento di una classe in PragmaDB

Per inserire una classe in PragmaDB è necessario applicare la seguente procedura:

- effettuare l'accesso in PragmaDB;
- selezionare la voce "Classi";
- selezionare la voce "Inserisci Classe";
- popolare i campi richiesti;
- inserire i requisiti correlati.

2.1.7.7 Tracciamento requisiti-fonti

Per tracciare una fonte su un requisito è sufficiente, al momento della creazione del requisito, inserire la fonte correlata. Se questo non è stato fatto, è necessario applicare la seguente procedura:

- effettuare l'accesso in PragmaDB;
- selezionare la voce "Requisiti";
- selezionare la voce "Modifica" del requisito interessato;
- selezionare e inserire la fonte correlata.

2.1.7.8 Tracciamento componenti-requisiti

Ancora non so!

2.1.7.9 Tracciamento classi-requisiti

Per tracciare un requisito su una classe è sufficiente, al momento della creazione della classe, inserire il requisito correlato. Se questo non è stato fatto, è necessario applicare la seguente procedura:

- effettuare l'accesso in PragmaDB;
- selezionare la voce "Classe";
- selezionare la voce "Modifica" della classe interessata;
- selezionare e inserire il requisito correlato.

2.1.7.10 altri tracciamenti x -y ?

2.1.7.11 Produzione automatica del documento "*Specifica Tecnica v1.0.0*"

2.1.7.12 Produzione automatica del documento "*Definizione di Prodotto v1.0.0*"

2.1.7.13 Produzione automatica del documento "*Analisi dei Requisiti v1.0.0*"

Il documento "*Analisi dei Requisiti v1.0.0*" è generato automaticamente da PragmaDB, il quale produrrà il relativo codice LaTeX da compilare. Per far ciò, è necessario seguire la seguente procedura:

- effettuare l'accesso in PragmaDB;
- selezionare la voce "Use Case";
- selezionare tutte le voci sotto "Esporta in LaTeX";
- tornare nella home page;
- selezionare la voce "Requisiti";
- selezionare tutte le voci sotto "Esporta in LaTeX";
- aprire il file `RP/Esterni/AnalisiDeiRequisiti_v1.0.0.pdf`;
- in quest'ultimo file includere i file .tex generati da pragmaDB con il comando

  `input{nomeFile}`

  seguendo questo ordine:

- useCase;
- requisiti;
- tracciamentoFontiRequisiti;
- tracciamentoRequisitiFonti;
- riepilogoRequisiti;

2.1.8 Strumenti

2.1.8.1 PragmaDB

PragmaDB è uno strumento *open source*$_g$ di tracciamento dei requisiti. Verrà quindi utilizzato per semplificare e automatizzare il più possibile l'attività di analisi dei requisiti e di progettazione. In particolare, una volta inseriti *casi d'uso*$_g$, attori, requisiti, fonti, classi e package, PragmaDB genera:

- il codice LaTeX relativo a *casi d'uso*$_g$ e requisiti in forma tabellare;
- i diagrammi *UML*$_g$ associati ai *casi d'uso*$_g$.

Essendo *open source*$_g$, questo strumento è stato adattato dal team Co.Code in base alle proprie necessità.

2.1.8.2 *Astah*$_g$

Astah$_g$ è uno strumento di modellazione *UML*$_g$. Qualora i diagrammi *UML*$_g$ generati da PragmaDB non siano soddisfacenti, si ricorrerà all'utilizzo di *Astah*$_g$. Viene utilizzata la versione 7.0 o superiori.

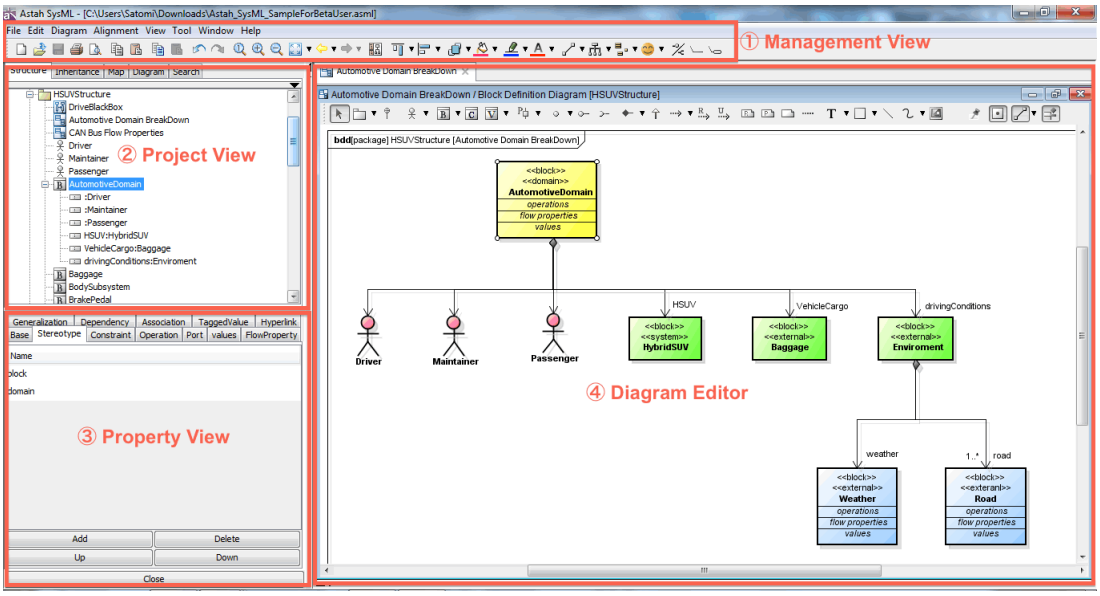

Figura 2: Astah

2.1.8.3 ...strumenti per il testing...

3 Processi di supporto

3.1 Documentazione

3.1.1 Scopo

Lo scopo di questo processo consiste nell'illustrazione di come deve essere redatta e mantenuta la documentazione, durante il *ciclo di vita*$_{\text{g}}$ del *software*$_{\text{g}}$.

3.1.2 Aspettative

Le aspettative della corretta implementazione di tale processo sono:

- una chiara visione della documentazione prodotta durante il *ciclo di vita*$_{\text{g}}$ del *software*$_{\text{g}}$;
- una serie di norme per la stesura di documenti coerenti e validi;
- una documentazione formale e coerente.

3.1.3 Descrizione

In questo documento devono essere redatte tutte le norme e le convenzioni adottate dal gruppo, in modo da produrre una documentazione valida e coerente.

3.1.4 Struttura dei documenti

3.1.4.1 Frontespizio

La prima pagina di ogni documento dovrà contenere:

- logo del gruppo;
- nome del *progetto*$_{\text{g}}$;
- nome del documento e la relativa versione;
- sommario;
- data di redazione;
- nome e cognome dei redattori del documento;
- nome e cognome dei verificatori del documento;
- nome e cognome del responsabile per l'approvazione del documento;
- uso del documento (interno o esterno);
- lista di distribuzione del documento.

3.1.4.2 Diario delle modifiche

La seconda pagina dovrà contenere il diario delle modifiche di quel determinato documento.
Il diario è costituito di una tabella ordinata in modo decrescente seconda la data di modifica e il numero di versione.
Gli attributi della tabella rappresentano:

- numero di versione;

- breve riepilogo delle modifiche apportate;
- autore delle modifiche;
- ruolo ricoperto dall'autore all'interno del *progetto*$_{g}$;
- data di modifica.

3.1.4.3 Indici

In ogni documento è presente un indice delle sezioni, utile a fornire una visione macroscopica della struttura del documento. Sono previsti, se necessari, gli indici relativi alle tabelle e alle figure presenti nel documento in questo ordine.

3.1.4.4 Intestazione e piè di pagina

L'intestazione delle pagine di ogni documento deve contenere:

- numero e titolo della sezione;
- nome del *progetto*$_{g}$.

Il piè di pagina contiene invece:

- nome del documento con la relativa versione;
- nome del gruppo;
- pagina X di Y, dove X è la pagina corrente e Y è il numero di pagine totali del documento.

3.1.4.5 Introduzione

In questa sezione vengono fornite le seguenti informazioni:

- scopo del documento;
- glossario;
- riferimenti utili;
  - riferimenti normativi;
  - riferimenti informativi.

3.1.5 Norme tipografiche

In questa sezione vengono definite le norme ortografiche e tipografiche da rispettare nella stesura di ogni documento.

3.1.5.1 Stile del testo

Al fine di migliorare la leggibilità e comprensione di un documento, è d'obbligo preferire uno stile d'esposizione sfruttando elenchi piuttosto che uno stile narrativo, in maniera tale da esporre i contenuti più esplicitamente.

- **Grassetto**: viene utilizzato per:
  - titoli;
  - elementi di un elenco puntato che riassumono il contenuto del relativo paragrafo.

- **Corsivo**: viene utilizzato per:
  - citazioni;
  - abbreviazioni;
  - parole inserite nel glossario;
  - riferimenti ad altri documenti;
  - nomi di società o aziende;
  - ruoli dei membri del gruppo.
- **Maiuscolo**: le parole scritte interamente in maiuscolo dovranno riferirsi soltanto ad acronimi.
- **Monospace**: le porzioni di testo scritte in monospace definiscono:
  - frammenti di codice;
  - comandi;
  - URL.
- **Glossario**: le parole che hanno un riferimento nel glossario sono in corsivo e hanno una 'g' a pedice.

3.1.5.1.1 Righe mal poste

Le righe mal poste sono quelle righe che coincidono con una delle seguenti descrizione:

- una riga di un paragrafo (o un titolo di livello superiore o inferiore) che inizia alla fine di una pagina;
- una riga di un paragrafo (o un titolo di livello superiore o inferiore) che finisce all'inizio di una pagina.

Per una questione di leggibilità, queste tipologie di righe devono essere evitate.
L'utilizzo del comando LaTeX `\newpage` aiuta a risolvere questo problema.

3.1.5.1.2 Virgolette

I vari tipi di virgolette devono essere usati nei seguenti modi:

- **Virgolette singole ' '**: devono essere utilizzate solo per racchiudere un singolo carattere;
- **Virgolette doppie " "**: devono essere utilizzate solo per racchiudere:
  - citazioni;
  - nomi di documenti;
  - voci di un menù;
  - voci di pulsanti da premere.

3.1.5.2 Elenchi puntati

Tutti gli elenchi puntati sono caratterizzati graficamente da un pallino nel primo livello, da una trattino nel secondo e da un asterisco nel terzo (automatizzato grazie al *template*$_g$ LaTeX creato). Ogni elemento deve terminare con il punto e virgola, a meno che non sia l'ultimo dell'elenco, in questo caso la frase va terminata con il punto. Ogni punto inizia con la minuscola, tranne nel caso in cui necessiti di una spiegazione: allora si utilizzerà la maiuscola.
Il seguente esempio vuole chiarire la differenziazione grafica tra i vari livelli di un elenco.

- primo livello;
  - secondo livello;
    - terzo livello;
  - secondo livello;
- primo livello.

3.1.5.3 Formati comuni

- **Date**:

  AAAA - MM - GG

  dove:
  - AAAA: rappresenta l'anno utilizzando 4 cifre;
  - MM: rappresenta il mese utilizzando 2 cifre;
  - GG: rappresenta il giorno utilizzando 2 cifre.

  Ad esempio, 2016-12-01 è una data scritta secondo le norme stabilite;
- **Orari**:

  HH:MM

  dove:
  - HH: rappresenta l'ora e può assumere valori da 0 a 23;
  - MM: rappresenta i minuti e può assumere valori da 0 a 59.

  Ad esempio, 15:05 è un orario scritto secondo le norme stabilite;
- **Nomi ricorrenti**:
  - **Ruoli di *progetto*$_g$**: ogni nome di un ruolo di *progetto*$_g$ deve essere scritto con la lettera iniziale maiuscola e con lo stile corsivo. Questo viene automatizzato utilizzando il comando `\CodiceRuolo`;
  - **Nomi propri**: ogni nome deve essere espresso nella forma "Nome Cognome";
  - **Nomi dei documenti**: ogni nome di documento viene scritto con lo stile corsivo, con l'iniziale di ogni parola maiuscola e con la versione corrente. Questo viene automatizzato richiamando il comando `\SiglaDocumentodoc`.
- **URL**: un URL dovrà essere scritto in azzurro ed essere un collegamento alla risorsa a cui punta e, per rendere automatizzato questo stile, si deve far utilizzo del comando `\url`.
  Ad esempio, https://www.google.it/ è scritto secondo le norme stabilite.

3.1.5.4 Sigle

E' previsto l'utilizzo di queste sigle:

- **AdR**: per "*Analisi dei Requisiti v1.0.0*";
- **PdP**: per "*Piano di Progetto v1.0.0*";
- **NdP**: per "*Norme di Progetto v1.0.0*";
- **SdF**: per "*Studio di Fattibilità v1.0.0*";
- **PdQ**: per "*Piano di Qualifica v1.0.0*";
- **ST**: per "*Specifica Tecnica v1.0.0*";
- **Gl**: per "*Glossario v1.0.0*";
- **DP**: per "*Definizione di Prodotto v1.0.0*";
- **Rp**: per *Responsabile*;
- **Am**: per *Amministratore*;
- **Pt**: per *Progettista*;
- **An**: per *Analista*;
- **Pm**: per *Programmatore*;
- **Ve**: per *Verificatore*;
- **RR**: per **Revisione dei requisiti**;
- **RP**: per **Revisione di progettazione**;
- **RQ**: per **Revisione di qualifica**;
- **RA**: per **Revisione di accettazione**.

L'utilizzo di queste sigle è permesso all'interno di:

- tabelle;
- diagrammi;
- immagini;
- didascalie.

3.1.5.5 Nomi

3.1.5.5.1 Ruoli

Quando si fa riferimento ad un ruolo di progetto bisogna adottare lo stile corsivo e la prima lettera deve essere maiuscola.
Per automatizzare ciò, si deve far utilizzo dei comandi LaTeX definiti in `Docs/template/Riferimenti.sty`.

3.1.5.5.2 Periodi

Quando si fa riferimento ad un periodo del progetto bisogna utilizzare i comandi LaTeX definiti in `Docs/template/Riferimenti.sty`.

3.1.6 Elementi grafici

3.1.6.1 Tabelle

Al fine di facilitarne la tracciabilità, le tabelle devono essere:

- in netto stacco rispetto i paragrafi che seguono e precedono;
- accompagnate da una didascalia;
- accompagnate da un numero incrementale;
- centrate nella pagina.

3.1.6.2 Immagini

Al fine di facilitarne la tracciabilità, le immagini devono essere:

- in netto stacco rispetto i paragrafi che seguono e precedono;
- accompagnate da una didascalia;
- accompagnate da un numero incrementale;
- centrate nella pagina;
- essere di formato PNG.

3.1.7 Classificazione dei documenti

3.1.7.1 Documenti informali

Tutti i documenti sono da ritenersi informali fino all'approvazione da parte del *Responsabile* di *progetto*$_{\text{g}}$, ed in quanto tali sono da considerarsi esclusivamente ad uso interno.

3.1.7.2 Documenti formali

Un documento viene definito formale quando viene validato dal *Responsabile* di *progetto*$_{\text{g}}$. Solo i documenti formali possono essere distribuiti all'esterno del gruppo. Per arrivare a tale stato il documento deve aver passato la *verifica*$_{\text{g}}$ e la *validazione*$_{\text{g}}$.

3.1.7.3 Glossario

Il glossario nasce dall'esigenza di chiarire il significato di parole che possono risultare ambigue all'interno di determinati contesti. Saranno quindi presenti parole che:

- trattano argomenti tecnici;
- possono creare delle ambiguità sul significato;
- rappresentano delle sigle.

La struttura deve avere queste caratteristiche:

- le parole devono essere in ordine alfabetico;
- ogni termine deve essere seguito da una spiegazione chiara e concisa, che non generi alcun tipo di ambiguità.

3.1.7.4 Verbali

Questo documento ha lo scopo di riassumere in modo formale le discussioni effettuate e le decisioni prese durante le riunioni. I verbali, come le riunioni, sono classificati in: interni ed esterni. In particolare i verbali esterni, essendo documenti ufficiali, devono essere redatti dal *Responsabile* di Progetto. I verbali interni, invece, dovranno essere redatti dagli *Amministratori*.
Ogni verbale dovrà essere denominato nel seguente modo:

Verbale_TipoVerbale_DataVerbale

dove:

- **TipoVerbale**: identifica se il verbale è riferito ad una riunione interna (I) o esterna (E);
- **DataVerbale**: identifica la data nella quale si è svolta la riunione relativa al verbale.

Nella parte introduttiva vengono specificate le seguenti informazioni:

- luogo di incontro;
- data di incontro;
- orario di inizio;
- orario di fine;
- durata dell'incontro;
- oggetto dell'incontro;
- partecipanti;
- segretario;
- segnalazioni varie.

Tutte le decisioni prese durante la riunione vengono identificate univocamente utilizzando questo formato:

DIX.Y per i verbali interni

DEX.Y per i verbali esterni

dove:

- **X**: rappresenta il numero di verbale redatto in ordine cronologico (inizia da 1);
- **Y**: rappresenta il numero della decisione all'interno di un singolo verbale (inizia da 1).

Inoltre vengono tracciate le decisioni in sospeso che verranno chiarite in verbali successivi. Il formato identificativo è lo stesso delle decisioni definitive, dove nel codice la D viene sostituita dalla S.

3.1.8 Procedure

Per la stesura della documentazione si è utilizzato il linguaggio LaTeX, si veda la sezione strumenti 3.1.9.

3.1.8.1 Approvazione dei documenti

La formalizzazione di un documento segue la seguente procedura:

1. il documento viene redatto da coloro che sono incaricati della sua stesura ed eventuale correzione di errori;
2. per ogni significativa modifica del documento, i *Verificatori* avranno il compito di controllare la presenza di errori o imprecisioni;
3. se i *Verificatori* riscontrano degli errori, dovranno notificarlo ai redattori del documento tramite una specifica $issue_{g}$, tornando così al punto 1, altrimenti, se completo, il documento viene consegnato al *Responsabile*;
4. il *Responsabile* di $progetto_{g}$ decide se approvare, e quindi formalizzare il documento, oppure se rifiutarlo comunicando la motivazione e le modifiche da apportare, tornando così al punto 1.

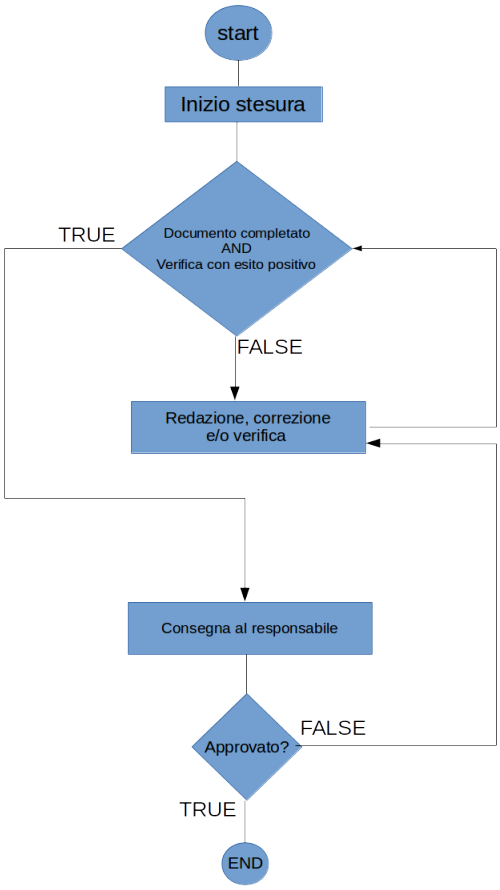

Figura 3: Flow chart dell'approvazione di un documento

3.1.8.2 Versionamento

Ciascun documento che verrà redatto dovrà essere versionato, per consentire un tracciamento chiaro della sua storia e delle sue modifiche.
Verrà applicato il seguente formalismo:

vX.Y.Z

dove:

- **X**:
  - inizia da 0;
  - viene incrementato quando il *Responsabile* di *progetto*$_{\text{g}}$ approva il documento.
- **Y**:
  - inizia da 0;
  - viene incrementato dal *Verificatore* ad ogni *verifica*$_{\text{g}}$;
  - quando viene modificato X, viene riportato a 0.
- **Z**:
  - inizia da 0;
  - viene incrementato dal Redattore del documento dopo ogni modifica;
  - quando viene modificato Y, viene riportato a 0.

3.1.8.3 Glossarizzazione

È stato creato uno script PHP che permette la glossarizzazione di tutti i documenti, ovvero marca in corsivo e con la g a pedice tutte le parole presenti nel "*Glossario v1.0.0*". Prima di ogni revisione, lo script dovrà essere lanciato da terminale con il comando

```
php glossarize2.php
```

3.1.8.4 Inserimento di un termine nel "*Glossario v1.0.0*"

Per inserire un termine nel "*Glossario v1.0.0*"è necessario seguire la seguente procedura:

- effettura l'accesso in PragmaDB;
- selezionare la voce "Glossario"
- inserire i campi richiesti.

3.1.8.5 Produzione automatica del documento "*Glossario v1.0.0*"

Per produrre automaticamente il "*Glossario v1.0.0*"è necessario seguire la seguente procedura:

- effettura l'accesso in PragmaDB;
- selezionare la voce "Glossario";
- selezionare tutte le voci sotto "Esporta in LaTeX";
- aprire il file chiamato `RP/Esterni/Glossario_v1.0.0.pdf`;
- includere il file .tex generato da PragmaDB con il seguente comando

`input{nomeFile}`

dove "nomeFile" è il nome del file generato a PragmaDB.

3.1.9 Strumenti

3.1.9.1 LaTeX

La stesura dei documenti deve essere effettuata utilizzando il linguaggio di *markup*$_g$ LaTeX. Le motivazioni di questa scelta sono dovute alle possibilità che LaTeX offre:

- creazione di documenti formali in modo rapido ed efficiente;
- possibilità di separare contenuto e formattazione, definendo l'aspetto delle pagine in un file *template*$_g$ separato e condiviso da tutti i documenti;
- creazione e gestione automatica dell'indice del documento.

3.1.9.1.1 Template

Per garantire omogeneità tra i documenti è stato creato un *template*$_g$ LaTeX, dove sono state definite tutte le regole di formattazione da applicare al documento. Questo permette a tutti i componenti del gruppo di concentrarsi solo nella stesura del contenuto, senza doversi preoccupare dell'aspetto.

3.1.9.1.2 Comandi personalizzati

Sono stati definiti dei comandi LaTeX personalizzati al fine di poter rendere più semplice ed immediata l'applicazione delle norme tipografiche. Questi comandi si occupano delle corretta formattazione del testo secondo le norme che sono state definite. La lista dei comandi è presente nel file `Docs/template/Comandi.sty`.

3.1.9.1.3 Riferimenti personalizzati

Sono stati definiti dei riferimenti LaTeX personalizzati al fine di poter rendere più semplice ed immediato il riferimento a ruoli, documenti, periodi e file. Questi riferimenti si occupano delle corretta formattazione del testo secondo le norme che sono state definite. La lista dei riferimenti è presente nel file `Docs/template/Riferimenti.sty`.

3.1.9.2 *Texmaker*$_g$

Per la redazione del codice LaTeX viene utilizzato l'editor *Texmaker*$_g$. Questo strumento oltre ad integrare un compilatore e visualizzatore *PDF*$_g$, fornisce suggerimenti per il completamento dei comandi LaTeX.

3.1.9.3 Excel

Per la creazione di grafici (istogrammi, diagrammi a torta, ecc.) viene utilizzato Excel di Microsoft Office, nella versione 2013 o successive.

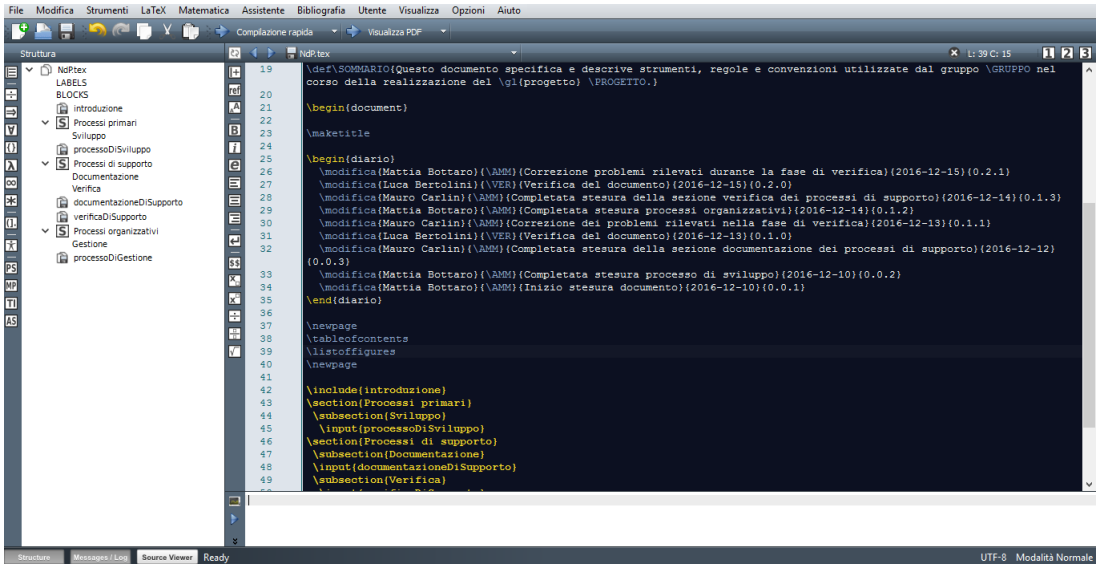

Figura 4: Texmaker

3.1.9.4 Script di glossarizzazione

In `Docs/script/glossarize2.php` si trova lo script che, per ogni documento, marca come glossarizzate le parole contenute in `Docs/template/terminiglossario.txt`, le quali poi saranno presenti nel "*Glossario v1.0.0*".

3.2 Qualità

3.2.1 Metriche

Per garantire la qualità del lavoro del team gli *Amministratori* hanno definito delle metriche, riportandole nel "*Piano di Qualifica v1.0.0*", che devono rispettare la seguente notazione:

M{X}{Y}{Z}

dove:

- **X** indica se la metrica si riferisce a prodotti o processi e può assumere i valori:
  - **PC** per indicare i processi;
  - **PD** per indicare i prodotti.
- **Y** presente solo se la metrica è riferita ai prodotti, indica se il termine *prodotto*$_g$ si riferisce a documenti o al software e può assumere i seguenti valori:
  - **D** per indicare i documenti;
  - **S** per indicare il software;
- **Z** indica il codice univoco della metrica (numero intero incrementale a partire da 1).

3.2.2 Obiettivi

Per garantire la qualità del lavoro del team, gli *Amministratori* hanno definito degli obiettivi di qualità, riportandoli nel "*Piano di Qualifica v1.0.0*", che devono rispettare la seguente notazione:

O{X}{Y}{Z}

dove:

- **X** indica se l'obiettivo si riferisce a prodotti o processi e può assumere i valori:
  - **PC** per indicare i processi;
  - **PD** per indicare i prodotti.
- **Y** presente solo se l'obiettivo è riferito ai prodotti, indica se il termine *prodotto*$_g$ si riferisce a documenti o al software e può assumere i seguenti valori:
  - **D** per indicare i documenti;
  - **S** per indicare il software;
- **Z** indica il codice univoco dell'obiettivo (numero intero incrementale a partire da 1).

3.2.3 Procedure

3.2.3.1 Calcolo dell'indice di Gulpease

Affinché un documento possa superare la fase di approvazione, è necessario che soddisfi il test di leggibilità con un indice Gulpease superiore a 40 punti. Per valutare questa metrica di qualità, è necessario seguire la seguente procedura:

- dirigersi con il terminale in `Docs/script/gulpease.php`;
- dare il comando `php gulpease.php`;
- visualizzare il risultato sul terminale.

3.2.3.2 Controllo ortografico

Per verificare la correttezza ortografica è necessario seguire la seguente procedura:

- aprire Texmaker;
- aprire il documento interessato nel formato .tex;
- dal menù a tendina "Modifica", selezionare la voce "verifica ortografia".

Per rendere ciò possibile, è necessario installare il pacchetto relativo dizionario italiano per Texmaker.

3.2.3.3 Resoconto stato metriche

Per visualizzare il resoconto corrente dello stato di ciò che le metriche indicano, è necessario seguire la seguente procedura:

- effettuare l'accesso in PragmaDB;
- selezionare la voce "Metriche".

3.2.4 Strumenti

3.2.4.1 Script per il calcolo dell'indice di Gulpease

In `Docs/script/gulpease.php` si trova lo script che calcola l'indice di Gulpease per ogni documento.

3.2.4.2 Controllo ortografico

Per il controllo ortografico dei documenti si farà utilizzo dello strumento integrato in Texmaker. Per poterlo utilizzare, è necessario disporre del pacchetto per il dizionario italiano.

3.2.4.3 Requisiti obbligatori soddisfatti

Lo strumento scelto per il calcolo del valore di questa metrica è PragmaDB, il quale permette di tracciare i requisiti ed associarli su use case e fonti.

3.2.4.4 Requisiti accettati soddisfatti

Lo strumento scelto per il calcolo del valore di questa metrica è PragmaDB, il quale permette di tracciare i requisiti ed associarli su use case e fonti.

3.2.4.5 Requisiti non accettati soddisfatti

Lo strumento scelto per il calcolo del valore di questa metrica è PragmaDB,

3.2.4.6 Requisiti obbligatori soddisfatti

Lo strumento scelto per il calcolo del valore di questa metrica è PragmaDB, il quale permette di tracciare i requisiti ed associarli su use case e fonti.

3.2.4.7 Structural Fan-In

Lo strumento scelto per il calcolo del valore di questa metrica è PragmaDB,

3.2.4.8 Structural Fan-Out

Lo strumento scelto per il calcolo del valore di questa metrica è PragmaDB,

3.2.4.9 Metodi per classe

Lo strumento scelto per il calcolo del valore di questa metrica è PragmaDB, il quale permette di tracciare le classi ed associarle ad altre classi correlate e requisiti.

3.2.4.10 Parametri per metodo

Lo strumento scelto per il calcolo del valore di questa metrica è PragmaDB,

3.2.4.11 Componenti integrate

Lo strumento scelto per il calcolo del valore di questa metrica è PragmaDB,

3.2.4.12 Test di unità eseguiti

Lo strumento scelto per il calcolo del valore di questa metrica è PragmaDB,

3.2.4.13 Test di integrazione eseguiti

Lo strumento scelto per il calcolo del valore di questa metrica è PragmaDB,

3.2.4.14 Test di sistema eseguiti

Lo strumento scelto per il calcolo del valore di questa metrica è PragmaDB,

3.2.4.15 Test di validazione eseguiti

Lo strumento scelto per il calcolo del valore di questa metrica è PragmaDB,

3.2.4.16 Test superati

Lo strumento scelto per il calcolo del valore di questa metrica è PragmaDB,

3.2.4.17 Completezza implementazione funzionale

Lo strumento scelto per il calcolo del valore di questa metrica è PragmaDB,

3.2.4.18 Densità di failure

Lo strumento scelto per il calcolo del valore di questa metrica è PragmaDB,

3.3 Configurazione

3.3.1 Versionamento

3.3.1.1 Repository

Per il versionamento e l'archiviazione dei file, l'*Amministratore* ha creato un *repository*$_g$ *GitHub*$_g$, il quale è disponibile al seguente indirizzo https://github.com/CoCodeSWE. Tutti i membri del gruppo dovranno creare un proprio account *GitHub*$_g$, per poi ricevere i permessi in scrittura sul *repository*$_g$ da parte dell'*Amministratore*. La gestione del *repository*$_g$ è responsabilità degli *Amministratori*.

3.3.1.2 Struttura del *repository*$_g$ Docs

Al fine di mantenere ordine e coerenza tra i file, il *repository*$_g$ è così strutturato:

- Docs
  - RR
    - Esterni: contiene i documenti esterni;
    - Interni: contiene i documenti interni.
  - RP
    - Esterni: contiene i documenti esterni;
    - Interni: contiene i documenti interni.
  - script: contiene gli script utilizzati;
  - *template*$_g$: contiene i *template*$_g$ utilizzati.

In futuro verranno aggiunte nuove *repository*$_g$, le quali strutture saranno rappresentate come precedentemente fatto con Docs.

3.3.1.3 Commit

Ogni commit effettuata deve essere accompagnata da un messaggio descrittivo delle modifiche effettuate. L'autore della commit dovrà assicurarsi della correttezza dei file. È sconsigliato effettuare il commit di intere cartelle, al fine di evitare inclusioni di file inutili(ad es: file di compilazione). Dovrà inoltre essere segnalata l'eventuale aggiunta di nuovi file.

3.3.1.4 Modifiche

In caso di modifiche importanti da attuare, si dovrà creare un *branch*$_g$ apposito nel quale lavorare, al fine di proteggere il branch principale di sviluppo. Una volta apportate, le modifiche dovranno essere applicate al branch principale.

3.3.1.5 Procedure

3.3.1.5.1 Aggiornamento del repository locale

Per aggiornare il repository in oggetto, è necessario seguire la seguente procedura:

- aprire il terminale nel repository;
- dare il comando `git pull`.

Se l'operazione non è possibile a causa di conflitti sui file, è necessario risolverli a mano per poi riapplicare la procedura precedente.

3.3.1.5.2 Aggiornamento del repository remoto

Per aggiornare il repository in oggetto con le proprie modifiche locali, è necessario seguire la seguente procedura:

- aprire il terminale nel repository;
- dare il comando `git add lista_file` ;
- dare il comando `git commit -m "messaggio"`, dove il messaggio dovrà essere un breve testo esplicativo delle modifiche che si stanno apportando;
- dare il comando `git push`.

3.3.1.5.3 Utilizzo di un branch

Per creare un branch è necessario seguire la seguente procedura:

- aprire il terminale nel repository;
- dare il comando `git checkout -b nomeBranch`.

Una volta dato `git push`, verrà creato il corrispondente branch in GitHub.
Una volta terminato il suo utilizzo, il branch dovrà essere eliminato seguendo questa procedura:

- aprire il terminale nel repository;
- se ci si trova sul branch da eliminare, spostarsi da esso con `git checkout master`;
- dare il comando `git checkout -d nomeBranch`.

3.3.1.6 Strumenti

3.3.1.6.1 Git

Lo strumento usato per il versionamento è *Git*$_g$, in combinazione con il servizio di host GitHub e con GitHub desktop. Git è un software open-source di controllo versione distribuito utilizzabile dal terminale. Come versione si utilizza la 2.7.4 o superiori.

3.3.1.6.2 GitHub

GitHub$_g$ è un servizio di hosting per progetti *software*$_g$, con il quale è possibile interagire tramite *Git*$_g$. *GitHub*$_g$ offre diversi piani per *repository*$_g$ privati sia a pagamento, sia gratuiti, molto utilizzati per lo sviluppo di progetti open-source.

3.3.1.6.3 GitHub desktop

GitHub$_g$ Desktop è l'applicativo desktop per contribuire e collaborare ai progetti del corrispondente servizio web *GitHub*$_g$. Esso è disponibile per Windows e MacOS. Per Windows si utilizza la versione 3.3.3 o superiori.

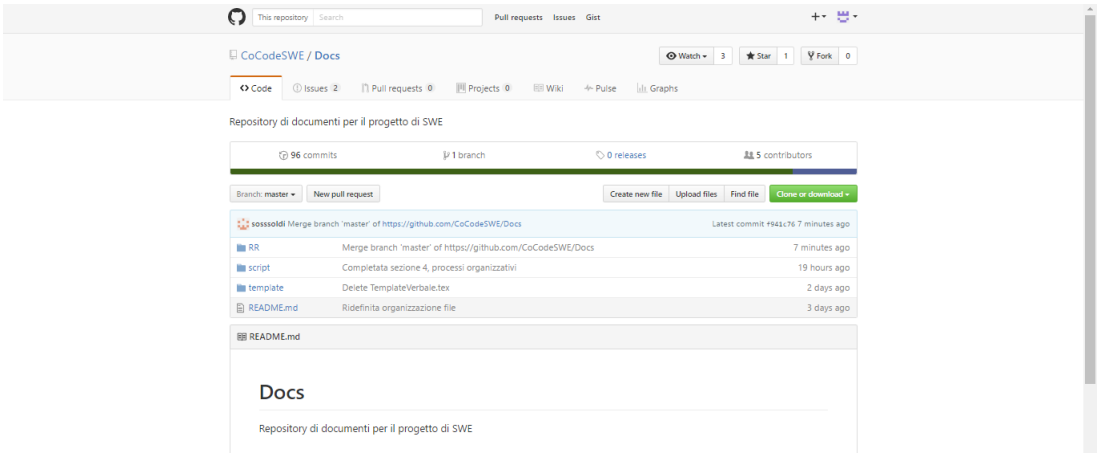

Figura 5: Github

3.4 *Verifica*$_g$

3.4.1 Scopo del processo

Si occupa di accertare che lo svolgimento del processo in esame non introduca errori nel *prodotto*$_g$.

3.4.2 Aspettative del processo

Una corretta implementazione di tale processo permette di individuare:

- una procedura di *verifica*$_g$;
- i criteri per la *verifica*$_g$ del *prodotto*$_g$.

3.4.3 Documenti

Il *Responsabile* ha il compito di avviare la fase di verifica, assegnando i task ai *Verificatori*. Essi dovranno controllare che:

- la sintassi sia corretta, facendo utilizzo degli strumenti automatici preposti e della tecnica walkthrough;
- i periodi non siano troppo lunghi, facendo utilizzo degli strumenti automatici preposti e della tecnica walkthrough;
- la struttura sia completa, non complicata e coerente con il contenuto del documento in esame.

3.4.4 Diagrammi UML

I *Verificatori* devono controllare tutti i diagrammi UML prodotti, sia che venga rispettato lo standard UML, sia che siano corretti semanticamente.

3.4.4.1 Diagrammi dei casi d'uso

Per tutti i diagrammi d'uso, i *Verificatori* dovranno controllare che:

- rispettino lo standard UML;

- rappresentino ciò che dovrebbero modellare, facendo attenzione a inclusione, estensioni e generalizzazione;
- gli attori correlati siano corretti;
- non ci siano difetti grafici.

3.4.4.2 Diagrammi di sequenza

Per tutti i diagrammi di sequenza, i *Verificatori* dovranno controllare che:

3.4.4.3 Diagrammi di attività

Per tutti i diagrammi di attività, i *Verificatori* dovranno controllare che:

3.4.4.4 Diagramma dei package

Per tutti i diagrammi dei package, i *Verificatori* dovranno controllare che:

3.4.5 Attività

3.4.5.1 Analisi statica

E' una tecnica di analisi del codice sorgente e della documentazione associata, prevalentemente usata quando il *sistema*$_{\text{g}}$ non è ancora disponibile e durante tutto l'arco del suo sviluppo. Non richiede l'esecuzione del *prodotto*$_{\text{g}}$ *software*$_{\text{g}}$ in alcuna sua parte. Può essere applicata tramite una delle seguenti strategie:

- **Walkthrough**: si legge l'intero documento (o codice) in cerca di tutte le possibili anomalie. E' una tecnica onerosa che richiede l'impegno di più persone e per questo deve essere utilizzata solo durante la prima parte del *progetto*$_{\text{g}}$, dove non tutti i membri hanno piena padronanza e conoscenza delle "*Norme di Progetto v1.0.0*"e del "*Piano di Qualifica v1.0.0*";
- **Inspection**: questa tecnica dev'essere applicata quando si ha idea della problematica che si sta cercando; consiste in una lettura mirata del documento (o del codice), sulla base di una lista degli errori precedentemente stilata.

3.4.5.2 Analisi dinamica

L'attività di analisi dinamica è una tecnica di *verifica*$_{\text{g}}$ applicabile solamente al *software*$_{\text{g}}$. Tale tecnica può essere utilizzata per analizzare l'intero *software*$_{\text{g}}$ o una porzione limitata dello stesso. L'attività consiste nell'esecuzione di test automatici realizzati dal team. Le verifiche devono essere effettuate su un insieme finito di casi, con valori di ingresso, uno stato iniziale e un esito decidibile. Tutti i test producono risultati automatici che inviano notifiche sulla tipologia di problema individuato. Ogni test è ripetibile, ossia applicabile durante l'intero *ciclo di vita*$_{\text{g}}$ del *software*$_{\text{g}}$.

3.4.6 Procedure

3.4.6.1 Issue tracking

L'*issue*$_{\text{g}}$ tracking è un'attività di supporto per la figura dei *Verificatori*, ai quali permette di tenere traccia, e contemporaneamente segnalare al *Responsabile*, la presenza di potenziali errori in un documento o nel codice sorgente.

3.4.6.1.1 Gestione delle *issue*$_g$

Qualora un *Verificatore* dovesse riscontrare delle anomalie, la procedura per la segnalazione e gestione del *ticketing*$_g$ di una *issue*$_g$ è la seguente:

1. il *Verificatore* dovrà aprire una nuova *issue*$_g$ assegnandole una label che si riferisca al problema trovato;
2. il *Responsabile* di *progetto*$_g$ dovrà valutare la *issue*$_g$; se la ritiene appropriata assegnerà ai redattori del documento (o ai *Programmatori*) il compito di risolvere la *issue*$_g$;
3. una volta risolta, e verificata, la *issue*$_g$ dovrà essere marcata come conlcusa da parte del *Responsabile* o del *Verificatore*;

3.4.7 Strumenti

3.4.7.1 Strumenti per l'issue tracking

Lo strumento utilizzato per l'*issue*$_g$ tracking è il servizio Issues messo a disposizione da *GitHub*$_g$.

3.4.7.2 Verifica ortografica

Viene utilizzata la *verifica*$_g$ in tempo reale dell'ortografia, integrata in TexMaker. Essa marca, sottolineando in rosso, le parole errate secondo la lingua italiana.

3.4.7.3 Indice di Gulpease

Affinché un documento possa superare la fase di approvazione, è necessario che soddisfi il test di leggibilità con un indice Gulpease superiore a 40 punti.

3.5 Validazione

3.5.1 Scopo

Lo scopo di questo processo è verificare il prodotto ottenuto sia coerente rispetto gli obiettivi prefissati.

3.5.2 Aspettative

Le aspettative di questo processo sono che:

- il prodotto finale rispetti i parametri di qualità imposti;
- il prodotto finale sia conforme rispetto le aspettative;
- il prodotto finale sia corretto.

3.5.3 Descrizione

Le responsabilità sono così distribuite:

- i *Verificatori* hanno il compito di eseguire i test, tracciandone i risultati;

- il *Responsabile* revisiona i risultati dei test, decidendo se ritenerli accettabili o meno. Inoltre, si assume la responsabilità con il committente della conformità del prodotto finale rispetto le aspettative.

3.5.4 Procedure

La procedura di validazione è composta dai seguenti punti:

- i *Verificatori* effettuano test manuali sul prodotto finale, tracciandone i risultati;
- il *Responsabile* esamina i risultati e decide se ritenerli buoni o se ripetere i test;
- se accettati, il *Responsabile* dovrà consegnare i risultati al proponente.

3.6 Qualità

3.6.1 Notazione

3.6.1.1 Metriche

Per garantire la qualità del lavoro del team gli *Amministratori* hanno definito delle metriche, riportandole nel "*Piano di Qualifica v1.0.0*", che devono rispettare la seguente notazione:

M{X}{Y}{Z}

dove:

- **X** indica se la metrica si riferisce a prodotti o processi e può assumere i valori:
  - **PC** per indicare i processi;
  - **PD** per indicare i prodotti.
- **Y** presente solo se la metrica è riferita ai prodotti, indica se il termine *prodotto*$_g$ si riferisce a documenti o al software e può assumere i seguenti valori:
  - **D** per indicare i documenti;
  - **S** per indicare il software;
- **Z** indica il codice univoco della metrica (numero intero incrementale a partire da 1).

3.6.1.2 Obiettivi

Per garantire la qualità del lavoro del team, gli *Amministratori* hanno definito degli obiettivi di qualità, riportandoli nel "*Piano di Qualifica v1.0.0*", che devono rispettare la seguente notazione:

O{X}{Y}{Z}

dove:

- **X** indica se l'obiettivo si riferisce a prodotti o processi e può assumere i valori:
  - **PC** per indicare i processi;
  - **PD** per indicare i prodotti.
- **Y** presente solo se l'obiettivo è riferito ai prodotti, indica se il termine *prodotto*$_g$ si riferisce a documenti o al software e può assumere i seguenti valori:
  - **D** per indicare i documenti;
  - **S** per indicare il software;
- **Z** indica il codice univoco dell'obiettivo (numero intero incrementale a partire da 1).

3.6.2 Definizione metriche

Di seguito sono definite le metriche utilizzate nel documento "*Piano di Qualifica v2.0.0*". Ad ogni metrica è stata assegnato un codice identificativo per facilitare il tracciamento.

3.6.2.1 Qualità di processo

3.6.2.1.1 Percentuale di accessi avvenuti correttamente a PragmaDB - MPC1

Indica il numero di accessi avvenuti correttamente espresso in percentuale.

$$\text{Percentuale accessi} = \frac{\text{Accessi avvenuti}}{\text{Richieste d'accesso}} * 100$$

3.6.2.1.2 Schedule Variance - MPC2

Indica se si è in linea, in anticipo o in ritardo rispetto ai tempi pianificati.

$$\text{Schedule Variance} = \text{TP - TR}$$

- **TP** è il tempo pianificato per terminare un attività;
- **TR** è il tempo reale che è stato impiegato.

3.6.2.1.3 Cost Variance in percentuale - MPC3

Indica se alla data corrente i costi corrispondono alla pianificazione, espressi in percentuale.

$$\text{Cost Variance} = \frac{CP - CR}{CP} * 100$$

- **CP** sono i costi pianificati per la data corrente;
- **CR** sono i costi reali sostenuti.

3.6.2.1.4 Indice dei rischi non preventivati - MPC4

É un indice che viene incrementato ogniqualvolta si manifesta un rischio non individuato nell'attività di analisi dei rischi.

3.6.2.1.5 Numero di requisiti obbligatori soddisfatti - MPC5

Indica la percentuale di requisiti obbligatori soddisfatti del prodotto.

$$Numerorequisiti = \frac{\text{requisiti obbligatori soddisfatti}}{\text{requisiti obbligatori totali}} * 100$$

3.6.2.1.6 Numero di requisiti desiderabili soddisfatti - MPC6

Indica la percentuale di requisiti desiderabili soddisfatti del prodotto.

$$\text{Numero requisiti} = \frac{\text{requisiti desiderabili soddisfatti}}{\text{requisiti desiderabili totali}} * 100$$

3.6.2.1.7 Numero di requisiti opzionali soddisfatti - MPC7

Indica la percentuale di requisiti opzionali soddisfatti del prodotto.

$$\text{Numero requisiti} = \frac{\text{requisiti opzionali soddisfatti}}{\text{requisiti opzionalii totali}} * 100$$

3.6.2.1.8 SF-IN - MPC8

Indice numerico che incrementa nel momento in cui viene individuato un modulo che, durante la sua esecuzione, chiama il modulo in oggetto.

3.6.2.1.9 SF-OUT - MPC9

Indice numerico che incrementa nel momento in cui viene individuato un modulo utilizzato dal modulo in oggetto durante la sua esecuzione.

3.6.2.1.10 ???? - MPC10

3.6.2.1.11 Numero di metodi per classe - MPC11

Indica il numero di metodi definiti in ogni classe.

3.6.2.1.12 Numero di parametri per metodo - MPC12

Indica il numero di parametri definiti in ogni metodo.

3.6.2.1.13 Indice di complessità ciclomatica - MPC13

Dato un grafo non fortemente connesso che rappresenta una sezione di codice del software, indica il numero di cammini linearmente indipendenti.

$$\text{Complessità ciclomatica} = E - N + 2P$$

- **E** è il numero di archi del grafo;
- **N** è il numero di nodi del grafo;
- **P** è il numero di componenti connesse.

3.6.2.1.14 Numero di livelli di annidamento - MPC14

Indica il numero di procedure e funzioni annidate, ovvero richiamate all'interno di altre procedure o funzioni.

3.6.2.1.15 Percentuale di linee di commento per linee di codice - MPC15

Indica la percentuale di linee di commento rispetto alle linee di codice.

$$\text{Percentuale linee di commento} = \frac{\text{Linee di commento}}{\text{Linee di codice}} * 100$$

3.6.2.1.16 Halstead Difficulty per funzione - MPC16

Indica il livello di complessità di una funzione.

$$\text{Halstead Difficulty} = \frac{UOP}{2} * \frac{OD}{UOD}$$

- **UOP** è il numero di operatori distinti;
- **OD** è il numero totale di operandi;
- **UOD** è il numero di operandi distinti.

3.6.2.1.17 Halstead Volume per funzione - MPC17

Indica la dimensione dell'implementazione di un algoritmo.

$$\text{Halstead Volume} = (OP + OD) * \log_2(UOP + UOD)$$

- **OP** è il numero totale di operatori;
- **UOP** è il numero di operatori distinti;
- **OD** è il numero totale di operandi;
- **UOD** è il numero di operandi distinti.

3.6.2.1.18 Halstead Effort per funzione - MPC18

Indica il costo necessario a scrivere il codice di una funzione.

$$\text{Halstead Effort} = \text{Halsted Difficulty} * \text{Halstead Volume}$$

3.6.2.1.19 Indice di manutenibilità - MPC19

Indica quanto sarà semplice mantenere il codice prodotto.

$$\text{Manutenibilità} = 171 - 5,2 * \ln(\text{Halstead Volume}) - 0.23 * (\text{Complessita ciclomatica}) - 16.2 * \ln(\text{Linee di codice})$$

3.6.2.1.20 Percentuale di componenti integrate nel sistema - MPC20

Indica la percentuale di componenti attualmente implementate e correttamente integrate nel sistema.

$$\text{Componenti integrate} = \frac{\text{Numero componenti integrate}}{\text{Numero componenti totali progettate}} * 100$$

3.6.2.1.21 Percentuale di test di unità eseguiti - MPC21

Indica la percentuale di test di unità eseguiti.

$$\text{Test di unità eseguiti} = \frac{\text{Numero test di unita eseguiti}}{\text{Numero test di unita pianificati}} * 100$$

3.6.2.1.22 Percentuale di test di integrazione eseguiti - MPC22

Indica la percentuale di test di integrazione eseguiti.

$$\text{Test di integrazione eseguiti} = \frac{\text{Numero test di integrazione eseguiti}}{\text{Numero test di integrazione pianificati}} * 100$$

3.6.2.1.23 Percentuale di test di sistema eseguiti - MPC23

Indica la percentuale di test di sistema eseguiti.

$$\text{Test di sistema eseguiti} = \frac{\text{Numero test di sistema eseguiti}}{\text{Numero test di sistema pianificati}} * 100$$

3.6.2.1.24 Percentuale di test di validazione eseguiti - MPC24

Indica la percentuale di test di validazione eseguiti.

$$\text{Test di validazione eseguiti} = \frac{\text{Numero test di validazione eseguiti}}{\text{Numero test di validazione pianificati}} * 100$$

3.6.2.1.25 Percentuale dei test superati - MPC25

Indica la percentuale di test superati.

$$\text{Test superati} = \frac{\text{Numero test superati}}{\text{Numero test eseguiti}} * 100$$

3.6.2.1.26 Percentuale di rami decisionali percorsi - MPC26

Indica la percentuale di rami decisionali percorsi dai test di unità utilizzati.

$$\text{Rami decisionali percorsi} = \frac{\text{Numero rami decisionali percorsi}}{\text{Numero rami decisionali totali}} * 100$$

3.6.2.1.27 Numero di funzioni chiamate nei test - MPC27

Indica il numero di funzioni chiamate nel test di unità utilizzato.

3.6.2.1.28 Numero di istruzioni nei test - MPC28

Indica il numero di istruzioni eseguite nel test di unità utilizzato.

3.6.2.2 Qualità di prodotto

3.6.2.2.1 Indice Gulpease - MPDD1

Permette di calcolare i livello di leggibilità e comprensibilià del documento.

$$\textit{Indice Gulpease} = 89 + \frac{300 * A + 10 * B}{C}$$

- **A** è il numero totale di frasi;
- **B** è il numero totale di lettere
- **C** è il numero totale di parole;

3.6.2.2.2 Completezza dell'implementazione funzionale - MPDS1

Permette di calcolare i livello di leggibilità e comprensibilià del documento.

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

- **FM** è il numero di funzionalità mancanti nell'implementazione;
- **FI** è il numero di funzionalità individuate nell'attività di analisi;

3.6.2.2.3 Percentuale di risultati concordi alle attese - MPDS2

Calcola quanti risultati sono concordi alle attese.

$$RC = (1 - \frac{N_{RD}}{N_{TE}}) \cdot 100$$

- **RD** è il numero di test che producono risultati discordanti rispetto alle attese;
- **TE** è il numero di test-case eseguiti;

3.6.2.2.4 Percentuale di operazioni illegali non bloccate - MPDS3

Calcola quante operazioni illegali non sono state bloccate.

$$I = \frac{N_{IE}}{N_{II}} \cdot 100$$

- **IE** è il numero di operazioni illegali effettuabili dai test;
- **II** è il numero di operazioni illegali individuate;

3.6.2.2.5 Percentuale failure su test-case - MPDS4

Calcola la percentuale di operazioni di testing che si sono concluse in failure.

$$F = \frac{N_{FR}}{N_{TE}} \cdot 100$$

- **FR** è il numero di failure rilevati durante l'attività di testing;
- **TE** è il numero di test-case eseguiti;

3.6.2.2.6 Numero di failure evitati - MPDS5

Calcola la percentuale di funzionalità in grado di gestire correttamente i fault che potrebbero verificarsi.

$$B = \frac{N_{FE}}{N_{ON}} \cdot 100$$

- **FE** è il numero di failure evitati durante i test effettuati;
- **ON** è il numero di test-case eseguiti che prevedono l'esecuzione di operazioni non corrette, causa di possibili failure;

3.6.2.2.7 Percentuale delle funzionalità comprese - MPDS6

Calcola la percentuale di operazioni comprese in modo immediato dall'utente, senza la consultazione del manuale.

$$C = \frac{N_{FC}}{N_{FO}} \cdot 100$$

- **FC** è il numero di funzionalità comprese in modo immediato dall'utente durante l'attività di testing del prodotto;
- **FO** è il numero di funzionalità offerte dal sistema;

3.6.2.2.8 Percentuale di funzionalità conformi alle aspettative - MPDS7

Calcola la percentuale di funzionalità offerte all'utente che rispettano le sue aspettative riguardo al comportamento del software.

$$C = (1 - \frac{N_{MFI}}{N_{MFO}}) \cdot 100$$

- **MFI** è il numero di messaggi e funzionalità che non rispettano le aspettative dell'utente;
- **MFO** è il numero di messaggi e funzionalità offerti dal sistema;

3.6.2.2.9 Tempo medio di risposta - MPDS8

Calcola il periodo temporale medio trascorso tra la richiesta al software di una determinata funzionalità e la risposta all'utente.

$$T_{RISP} = \frac{\sum_{i=1}^{n} T_i}{n}$$

- T_{RISP} espresso in *secondi*;
- 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;

3.6.2.2.10 Percentuale di failure con cause individuate - MPDS9

Calcola la percentuale di failure di cui sono state individuate le cause.

$$I = \frac{N_{FI}}{N_{FR}} \cdot 100$$

- **FI** è il numero di failure delle quali sono state individuate le cause;
- **FR** è il numero di failure rilevate;

3.6.2.2.11 percentuale di failure introdotte con modifiche - MPDS10

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

$$I = \frac{N_{FRF}}{N_{FR}} \cdot 100$$

- **FRF** è il numero di failure risolte con l'introduzione di nuove failuree;
- **FR** è il numero di failure risolte;

3.6.3 Procedure

3.6.3.1 Calcolo dell'indice di Gulpease

Affinché un documento possa superare la fase di approvazione, è necessario che soddisfi il test di leggibilità con un indice Gulpease superiore a 40 punti. Per valutare questa metrica di qualità, è necessario seguire la seguente procedura:

- dirigersi da terminale in `Docs/script/gulpease.php`;

- dare il comando `php gulpease.php`;
- visualizzare il risultato sul terminale.

3.6.3.2 Controllo ortografico

Per verificare la correttezza ortografica è necessario seguire la seguente procedura:

- aprire Texmaker;
- aprire il documento interessato nel formato .tex;
- dal menù a tendina "Modifica", selezionare la voce "verifica ortografia".

Per rendere ciò possibile, è necessario installare il pacchetto relativo dizionario italiano per Texmaker.

3.6.3.3 Resoconto stato metriche

Per visualizzare il resoconto corrente dello stato di ciò che le metriche indicano, è necessario seguire la seguente procedura:

- effettuare l'accesso in PragmaDB;
- selezionare la voce "Metriche".

3.6.3.4 Resoconto stato test di unità

Per visualizzare il resoconto corrente dello stato dei test di unità è necessario seguire la seguente procedura:

- entrare nella cartella `/responsoTest`;
- aprire il file corrispondente al test eseguito, identificato con data e ora dell'esecuzione.

3.6.3.5 Resoconto stato build

Per visualizzare il resoconto corrente dello stato di una delle build eseguite, è necessario seguire la seguente procedura:

- entrare nella cartella `/statoBuild`;
- aprire il file corrispondente al numero della build sulla quale si vogliono informazioni.

3.6.4 Strumenti

3.6.4.1 Script per il calcolo dell'indice di Gulpease

In `Docs/script/gulpease.php` si trova lo script che calcola l'indice di Gulpease per ogni documento.

3.6.4.2 Controllo ortografico

Per il controllo ortografico dei documenti si farà utilizzo dello strumento integrato in Texmaker. Per poterlo utilizzare, è necessario disporre del pacchetto per il dizionario italiano.

3.6.4.3 Integrazione continua - Jenkins

Jenkins è uno strumento open source di continuous integration, scritto in linguaggio Java che fornisce dei servizi di integrazione continua per lo sviluppo del software. Viene eseguito lato server all'interno di un server web che supporta la tecnologia Servlet e quindi può essere utilizzato da remoto all'interno di un Web browser.
Questo strumento permetterà al gruppo di automatizzare i test di unità, che verranno eseguiti ad ogni push eseguita da un programmatore, ed il processo di build e deployment del prodotto.

3.6.4.3.1 Codifica ed esecuzione test di unità - Mocha

Mocha è un framework che permette di creare ed eseguire test sia per Node.js che per il browser web.

3.6.4.3.2 Calcolo della copertura dei test - Istanbul

Istanbul è uno strumento che calcola la copertura dei test rispetto a statement, righe, funzioni e branch di un determinato progetto.

3.6.4.4 Requisiti obbligatori soddisfatti

Lo strumento scelto per il calcolo del valore di questa metrica è PragmaDB, il quale permette di tracciare i requisiti ed associarli su use case e fonti.

3.6.4.5 Requisiti accettati soddisfatti

Lo strumento scelto per il calcolo del valore di questa metrica è PragmaDB, il quale permette di tracciare i requisiti ed associarli su use case e fonti.

3.6.4.6 Requisiti non accettati soddisfatti

Lo strumento scelto per il calcolo del valore di questa metrica è PragmaDB, il quale permette di tracciare i requisiti ed associarli su use case e fonti.

3.6.4.7 Requisiti obbligatori soddisfatti

Lo strumento scelto per il calcolo del valore di questa metrica è PragmaDB, il quale permette di tracciare i requisiti ed associarli su use case e fonti.

3.6.4.8 Structural Fan-In

Lo strumento scelto per il calcolo del valore di questa metrica è PragmaDB, il quale permette di tracciare quanti moduli durante la loro esecuzione utilizzano un determinato modulo.

3.6.4.9 Structural Fan-Out

Lo strumento scelto per il calcolo del valore di questa metrica è PragmaDB, il quale permette di tracciare quanti moduli vengono utilizzate durante l'esecuzione di un determinato modulo.

3.6.4.10 Metodi per classe

Lo strumento scelto per il calcolo del valore di questa metrica è PragmaDB, il quale permette di tracciare il numero di metodi per ogni classe.

3.6.4.11 Parametri per metodo

Lo strumento scelto per il calcolo del valore di questa metrica è PragmaDB, il quale permette di tracciare il numero di parametri per ogni metodo.

3.6.4.12 Componenti integrate

Lo strumento scelto per il calcolo del valore di questa metrica è PragmaDB, il quale permette di tracciare le componenti che sono attualmente implementate e integrate nel sistema.

3.6.4.13 Test di unità eseguiti

Lo strumento scelto per il calcolo del valore di questa metrica è PragmaDB, il quale permette di tracciare i test di unità eseguiti.

3.6.4.14 Test di integrazione eseguiti

Lo strumento scelto per il calcolo del valore di questa metrica è PragmaDB, il quale permette di tracciare i test di integrazione eseguiti.

3.6.4.15 Test di sistema eseguiti

Lo strumento scelto per il calcolo del valore di questa metrica è PragmaDB, il quale permette di tracciare i test di sistema eseguiti.

3.6.4.16 Test di validazione eseguiti

Lo strumento scelto per il calcolo del valore di questa metrica è PragmaDB, il quale permette di tracciare i test di validazione eseguiti.

3.6.4.17 Test superati

Lo strumento scelto per il calcolo del valore di questa metrica è PragmaDB, il quale permette di tracciare i test superati.

3.6.4.18 Completezza implementazione funzionale

Lo strumento scelto per il calcolo del valore di questa metrica è PragmaDB, il quale permette di tracciare il numero di requisiti funzionali implementati.

3.6.4.19 Densità di failure

Lo strumento scelto per il calcolo del valore di questa metrica è PragmaDB, il quale permette di tracciare le operazioni di testing che sono concluse in failure.

4 Processi organizzativi

4.1 Gestione

4.1.1 Scopo

Lo scopo del processo è produrre il "*Piano di Progetto v1.0.0*", al fine di pianificare e gestire i ruoli che i membri dovranno assumere.

4.1.2 Aspettative

Le aspettative del processo sono:

- produrre il "*Piano di Progetto v1.0.0*";
- definire i ruoli dei membri del gruppo;
- definire il piano per l'esecuzione dei compiti programmati.

4.1.3 Descrizione

4.1.4 Ruoli di progetto

In ogni momento temporale ogni membro deve ricoprire almeno un ruolo e, durante tutta la durata del $progetto_{\mathrm{g}}$, ricoprire tutti i ruoli almeno una volta. Per ogni membro, le ore di lavoro devono essere il più possibile equamente distribuite. L'assegnazione e la rotazione dei ruoli sono pianificate nel "*Piano di Progetto v1.0.0*".

4.1.4.1 Responsabile

Il *Responsabile* è il rappresentante e il punto di riferimento del gruppo, nonché colui che si assume le responsabilità delle scelte del gruppo. Le responsabilità assunte sono:

- pianificazione, coordinamento e controllo delle attività;
- gestione delle risorse;
- analisi e gestione dei rischi;
- approvazione dei documenti;
- approvazione dell'offerta economica;
- convocazione delle riunioni interne;
- relazioni esterne;
- assegnazione delle attività a persone.

Per questi motivi ha il compito di:

- assicurarsi che le attività di verifica e validazione siano svolte seguendo le normediprogetto;
- garantire il rispetto dei ruoli e dei compiti assegnati nel pianodiprogetto.

4.1.4.2 Amministratore

L'*Amministratore* è responsabile dell'efficienza dell'ambiente di lavoro, in particolare si occupa di:

- studiare e fornire strumenti che migliorano l'ambiente di lavoro, automatizzando il lavoro ove possibile;
- gestire archiviazione, versionamento e configurazione dei documenti e del *software*$_{\text{g}}$;
- garantire la qualità del *prodotto*$_{\text{g}}$, fornendo procedure e strumenti di monitoraggio e segnalazione;
- eliminare le difficoltà sulla gestione di processi e risorse.

L'*Amministratore* non compie scelte gestionali, ma tecnologiche concordate con il *Responsabile*.

4.1.4.3 Analista

L'*Analista* deve identificare e comprendere il dominio del problema.
In particolare si occupa di:

- mappare le richieste del cliente in specifiche per il *prodotto*$_{\text{g}}$;
- catalogare e spiegare specifiche comprensibili nell'"*Analisi dei Requisiti v1.0.0*" e nello "*Studio di Fattibilità v1.0.0*";
- classificare i requisiti;
- stendere i diagrammi dei casi d'uso;
- assegnare i requisiti a parti distinte del sistema;
- assicurarsi che i requisiti trovati siano conformi alle richieste del proponente;
- definire test di sistema e di accettazione al fine di verificare i requisiti.

L'*Analista* non si occupa di trovare una soluzione al problema, ma lo definisce redigendo lo *"Studio di Fattibilità"* e l'*"Analisi dei Requisiti"*.

4.1.4.4 Progettista

Il *Progettista* ha forti competenze sullo *stack*$_{\text{g}}$ tecnologico usato.
In particolare deve:

- indicare le tecnologie più adatte allo sviluppo del *progetto*$_{\text{g}}$;
- descrivere il funzionamento del *sistema*$_{\text{g}}$ progettandone l'architettura;
- produrre una soluzione fattibile in termini di risorse.

Il *Progettista* redige i documenti di *"Specifica Tecnica"*, *"Definizione di prodotto"* e si occupa delle sezioni del *"Piano di Qualifica"* relative alle metriche di verifica della programmazione.

4.1.4.5 Programmatore

Il *Programmatore* si occupa della codifica, in particolare:

- implementa le soluzioni indicate dal *Progettista*;
- scrive codice documentato, versionato e mantenibile nel rispetto delle "*Norme di Progetto v1.0.0*";
- realizza e fornisce gli strumenti per verificare e validare il *prodotto*$_g$.

4.1.4.6 Verificatore

Il *Verificatore*, disponendo di una profonda conoscenza delle "*Norme di Progetto v1.0.0*", si occupa delle attività di *verifica*$_g$.
In particolare deve:

- controllare il rispetto delle "*Norme di Progetto v1.0.0*"durante ogni attività del *progetto*$_g$.

4.1.4.7 Rotazione dei ruoli

Ogni membro del gruppo dovrà ricoprire ciascuno dei ruoli del progetto. La pianificazione dovrà essere redatta prestando attenzione a quanto segue:

- ogni membro del gruppo non dovrà mai ricoprire un ruolo che preveda la verifica dell'operato svolto da lui in precedenza poiché questo potrebbe portare ad un conflitto di interesse;
- bisogna tener conto dei possibili impegni o interessi dei singoli membri del gruppo;
- ciascun membro dovrà assicurare l'esclusivo svolgimento del ruolo a lui assegnato.

4.1.5 Comunicazioni

4.1.5.1 Interne

È stato creato un gruppo Telegram, accessibile solo ai membri del team, per effettuare le comunicazioni interne. In caso siano necessaria maggiore interazione, si farà utilizzo di *Google Hangouts*$_g$.

4.1.5.2 Esterne

È stata creata un'apposita cartella di posta elettronica per mantenere i contatti con il *proponente*$_g$, il committente ed altre eventuali figure esterne.
Inoltre, su richiesta dell'azienda proponente Zero12, è stato creato un apposito dominio Slack per permettere una veloce interazione tra i gruppi fornitori e l'azienda. Il dominio è zero12university.
La gestione della casella di posta elettronica e di Slack è compito del *Responsabile*.
L'indirizzo e-mail è il seguente: swe.co.code@gmail.com.

4.1.5.3 Email

Le mail devono essere scritte nella maniera più chiara e corretta possibile, in particolare ognuna di esse deve essere composta di:

- destinatario;
- oggetto, che deve essere breve e diretto;

- contenuto, che deve essere chiaro ed esaustivo;
- firma del responsabile.

Nel caso si vogliano scambiare dei documenti, si deve a evitare l'invio di allegati tramite email e preferire l'utilizzo di un apposito sito di scambi, quale ad esempio Google Drive.

4.1.6 Incontri

4.1.6.1 Interni

Ogni membro del team può proporre un incontro interno tramite il bot_{g} Telegram $VotePoll_{g}$, specificando i motivi e l'oggetto dell'incontro. Sarà poi compito del *Responsabile* decidere se effettuare l'incontro o meno.
La verbalizzazione degli incontri interni è compito di uno tra gli *Amministratori*.

4.1.6.2 Esterni

Ogni membro del team può proporre un incontro esterno tramite il Bot_{g} Telegram "$VotePoll_{g}$", specificando i motivi e l'oggetto dell'incontro. Se il *Responsabile* decide che l'incontro può essere organizzato dovrà accordarsi con la figura esterna, e comunicare gli estremi della riunione ai membri del team.
La verbalizzazione degli incontri esterni è compito del *Responsabile*.

4.1.6.3 Gestione

All'inizio di ogni riunione interna il *Responsabile* nomina l'amministratore che ha il compito di tracciare gli aspetti più importanti della riunione, con lo scopo di esporli poi nel relativo verbale. Questo verbale verrà archiviato nel repository del gruppo, in modo da permetterne la consultazione. Se invece la riunione è esterna, questo compito è delegato al *Responsabile*. Durante le riunioni i partecipanti devono tenere un comportamento che favorisca la discussione all'interno del gruppo.

4.1.7 Strumenti di coordinamento

4.1.7.1 $Ticketing_{g}$

Il *Responsabile* ha il compito di assegnare i $task_{g}$ ai membri del team utilizzando l'applicativo web $Asana_{g}$.
Definendo delle $milestone_{g}$, è possibile tenere traccia dello stato di avanzamento del lavoro di ogni $task_{g}$.

4.1.8 Rischi

Il *Responsabile* ha il dovere di individuare e monitorare i rischi indicati nel "*Piano di Progetto v1.0.0*". In caso ne vengano identificati di nuovi, il *Responsabile* deve agire nel modo seguente:

- comunicare i nuovi rischi al team;
- pianificare una strategia per la gestione dei nuovi rischi;
- aggiornare le procedure di gestione dei rischi nel "*Piano di Progetto v1.0.0*".

4.1.9 Procedure

4.1.9.1 Richiesta riunione interna

Per chiedere una riunione interna, si deve seguire la seguente procedura:

- effettuare l'accesso in Telegram;
- selezionare la chat "VoteBot";
- inserire in un solo messaggio:
  - oggetto della riunione;
  - contenuto da affrontare;
  - giorno proposto;
  - dare "Invio";
- inserire le opzioni di voto, le quali dovranno essere gli orari nei quali si propone di fare la riunione.
- selezionare "Publish Poll";
- selezionare la chat del gruppo.

Una volta che tutti hanno votato, il *Responsabile* provvederà ad approvare la proposta o meno.

4.1.9.2 Richiesta riunione esterna

Per chiedere una riunione esterna, si deve seguire la seguente procedura:

- effettuare l'accesso in Telegram;
- selezionare la chat "VoteBot";
- inserire in un solo messaggio:
  - oggetto della riunione;
  - contenuto da affrontare;
  - giorno proposto;
  - dare "Invio";
- inserire le opzioni di voto, le quali dovranno essere gli orari nei quali si propone di fare la riunione.
- selezionare "Publish Poll";
- selezionare la chat del gruppo.

Una volta che tutti hanno votato, il *Responsabile* provvederà ad approvare la proposta o meno.
Se approvata, il *Responsabile* provvederà a comunicare all'entità esterna gli estremi della riunione che il gruppo propone. In caso l'entità esterna proponga altri estremi, il *Responsabile* dovrà informare i membri del gruppo, al fine di decidere la fattibilità della proposta.

4.1.9.3 Gestione dei ticket

4.1.9.3.1 Creazione

Per creare un task viene utilizzato l'applicativo web Asana. Per ogni task viene creato un progetto separato rispetto al progetto AtAVi, col nome del task da eseguire. Tale progetto viene diviso in una serie di sottocompiti i quali verranno scritti come task sulla "bacheca" del progetto, senza essere ancora assegnati a nessuno.

4.1.9.3.2 Assegnazione

Per l'assegnazione dei task i membri del progetto si occuperanno di prendere un task a testa e di portarlo a termine, assegnandolo a sè stessi. Ogni volta che un membro porta a termine un task, ne prende un altro. I task saranno decisi dal responsabile in carica alla creazione del progetto, il quale dovrà anche controllare l'assegnazione dei compiti. A tale scopo, il responsabile sarà iscritto a tutti i compiti di tutti i progetti con l'account del gruppo, per facilitare il passaggio da un responsabile al successivo.

4.1.10 Strumenti

4.1.10.1 Telegram

Telegram è un *software*$_{\text{g}}$ libero che fornisce un servizio di messaggistica istantanea erogato senza fini di lucro dalla società Telegram LLC. È stato ritenuto più adatto di Whatsapp.

4.1.10.2 Google Hangouts

Hangouts è un *software*$_{\text{g}}$ di messaggistica istantanea e di *VoIP*$_{\text{g}}$ sviluppato da Google. È disponibile per le piattaforme mobili *Android*$_{\text{g}}$ e *iOS*$_{\text{g}}$ e come estensione per il *browser*$_{\text{g}}$ web Google Chrome. Inoltre, permette la condivisione degli schermi tra i membri della chiamata. È stato ritenuto più adatto di Skype.

4.1.10.3 Google Drive

Google Drive è un applicativo web che consente lo scambio di file tra più persone. Il gruppo Co.Codedovrà far utilizzo di questo strumento per lo scambio di file con le entità esterne.

4.1.10.4 Asana

Asana$_{\text{g}}$ è un applicativo web e mobile che consente al team di assegnare, tracciare e gestire dei *task*$_{\text{g}}$.

4.1.10.5 GanttProject

GanttProject è un *software*$_{\text{g}}$ gratuito per la creazione di grafici rappresentanti l'organizzazione e gestione di compiti e *milestone*$_{\text{g}}$ all'interno di un *progetto*$_{\text{g}}$. Verrà utilizzato nella versione 2.7 o superiore.

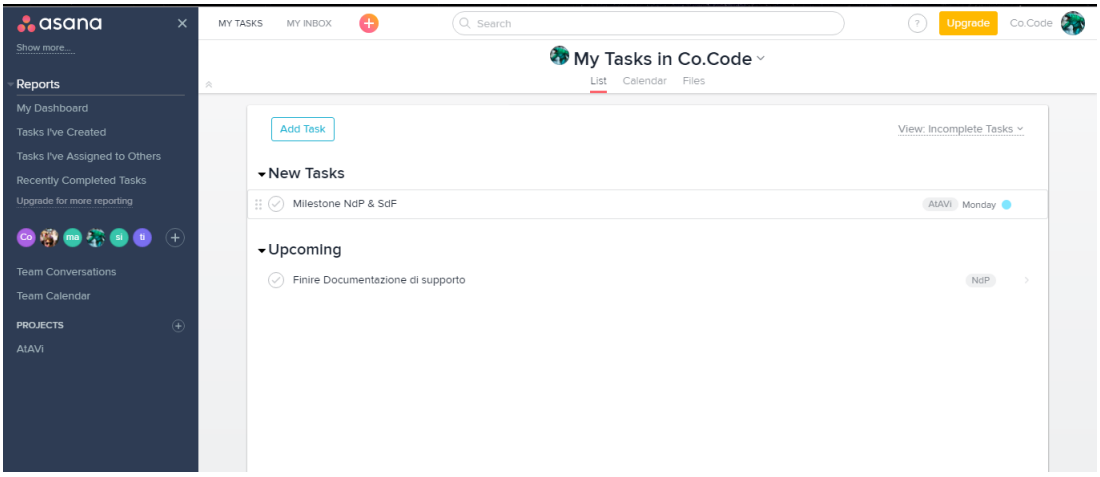

Figura 6: Asana

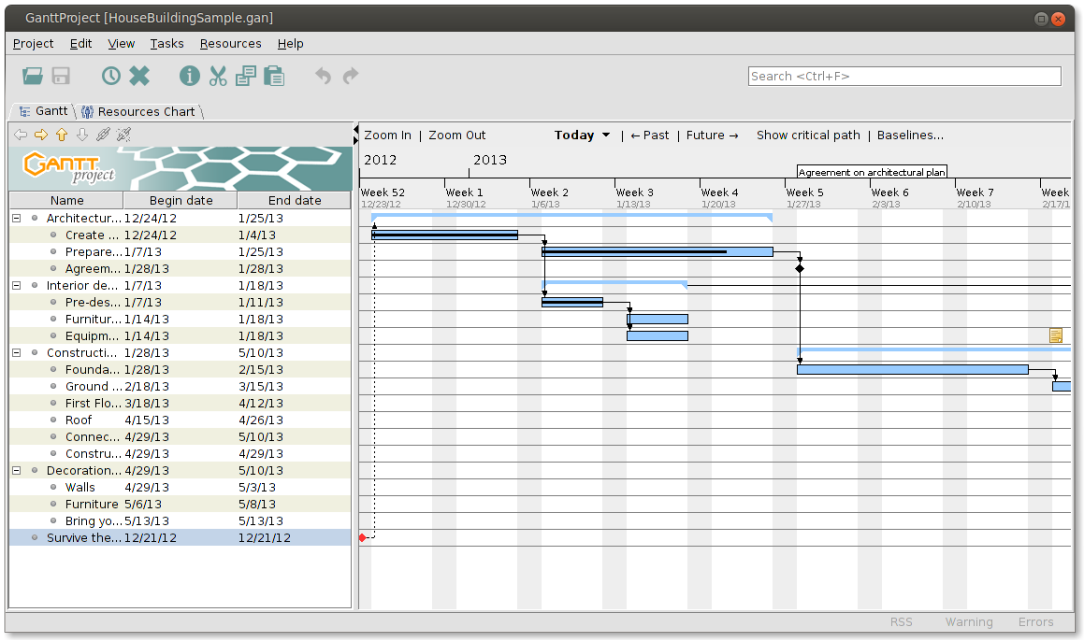

Figura 7: GanttProject