



# Agents of S.W.E.

A SOFTWARE COMPANY

Agents of S.W.E - Progetto "Plugin Grafana"

## Norme di Progetto

|                     |   |
|---------------------|---|
| <b>Versione</b>     | 0.0.5   |
| <b>Approvazione</b> | ?   |
| <b>Redazione</b>    | Luca Violato<br>Bogdan Stanciu<br>Marco Favaro<br>Marco Chilese     |
| <b>Verifica</b>     | ?<br>?  |
| <b>Stato</b>        | Work in Progress  |
| <b>Uso</b>          | Interno   |
| <b>Destinato a</b>  | Agents of S.W.E.<br>Prof. Tullio Vardanega<br>Prof. Riccardo Cardin |

agentsofswe@gmail.com



# Indice

|          |  |          |
|----------|--|----------|
| <b>1</b> | <b>Introduzione</b>                          | <b>1</b> |
| 1.1      | Scopo del Documento . . . . .                | 1        |
| 1.2      | Ambiguità e Glossario . . . . .              | 1        |
| 1.3      | Riferimenti . . . . .                        | 1        |
| <b>2</b> | <b>Processi Primari</b>                      | <b>2</b> |
| 2.1      | Fornitura . . . . .                          | 2        |
| 2.2      | Studio di fattibilità . . . . .              | 2        |
| 2.3      | Sviluppo . . . . .                           | 2        |
| 2.3.1    | Analisi dei requisiti . . . . .              | 2        |
| 2.3.1.1  | Classificazione dei requisiti . . . . .      | 2        |
| 2.3.1.2  | Classificazione dei casi d'uso . . . . .     | 2        |
| 2.3.2    | Progettazione . . . . .                      | 2        |
| 2.3.3    | Codifica . . . . .                           | 2        |
| 2.3.3.1  | Convenzioni per i nomi: . . . . .            | 2        |
| 2.3.3.2  | Convenzioni per la documentazione: . . . . . | 3        |
| 2.3.3.3  | ECMAScript 6: . . . . .                      | 3        |
| 2.3.3.4  | HTML . . . . .                               | 9        |
| <b>3</b> | <b>Processi di supporto</b>                  | <b>9</b> |
| 3.1      | Documentazione . . . . .                     | 9        |
| 3.1.1    | Descrizione . . . . .                        | 9        |
| 3.1.2    | Ciclo di vita documentazione . . . . .       | 9        |
| 3.1.3    | Template . . . . .                           | 9        |
| 3.1.4    | Struttura documenti . . . . .                | 9        |
| 3.1.4.1  | Prima pagina . . . . .                       | 9        |
| 3.1.4.2  | Piè di pagina . . . . .                      | 10       |
| 3.1.4.3  | Nomenclatura . . . . .                       | 10       |
| 3.1.4.4  | Tabelle . . . . .                            | 10       |
| 3.1.4.5  | Indice Sezioni . . . . .                     | 10       |
| 3.1.5    | Documenti Correnti . . . . .                 | 10       |
| 3.1.6    | Norme . . . . .                              | 10       |
| 3.1.6.1  | Struttura dei documenti . . . . .            | 10       |
| 3.1.6.2  | Norme tipografiche . . . . .                 | 10       |
| 3.1.7    | Struttura documentazione . . . . .           | 11       |
| 3.1.8    | Gestione termini Glossario . . . . .         | 11       |



|         |  |    |
|---------|--|----|
| 3.1.9   | Ambiente                                   | 11 |
| 3.2     | Qualità                                    | 11 |
| 3.2.1   | Descrizione                                | 11 |
| 3.2.2   | Classificazione dei processi               | 11 |
| 3.2.3   | Procedure                                  | 11 |
| 3.3     | Versionamento                              | 11 |
| 3.3.1   | Controllo di versione                      | 11 |
| 3.3.1.1 | Struttura del repository                   | 11 |
| 3.3.1.2 | Processo di implementazione                | 11 |
| 3.3.1.3 | Ciclo di vita dei branch                   | 12 |
| 3.3.1.4 | Rilascio di versione                       | 12 |
| 3.3.2   | Configurazione versionamento               | 12 |
| 3.3.2.1 | Remoto                                     | 12 |
| 3.3.2.2 | Locale                                     | 12 |
| 3.4     | Gestione di progetto                       | 13 |
| 3.4.1   | Configurazione strumenti di organizzazione | 13 |
| 3.4.1.1 | Inizializzazione                           | 13 |
| 3.4.1.2 | Aggiunta tasks                             | 13 |
| 3.4.1.3 | Aggiunta milestones                        | 13 |
| 3.4.2   | Ciclo di vita delle tasks                  | 13 |
| 3.4.2.1 | Apertura                                   | 13 |
| 3.4.2.2 | Completamento                              | 13 |
| 3.4.2.3 | Richiesta di revisione                     | 13 |
| 3.4.2.4 | Completamento                              | 13 |
| 3.5     | Verifica                                   | 13 |
| 3.5.1   | Descrizione                                | 13 |
| 3.5.2   | Analisi Statica                            | 13 |
| 3.5.3   | Verifica Diagrammi UML                     | 13 |
| 3.5.4   | Strumenti                                  | 13 |
| 3.6     | Validazione                                | 13 |
| 3.6.1   | Descrizione                                | 13 |
| 3.6.2   | Procedure                                  | 13 |
| 3.7     | Risoluzione problematiche                  | 13 |
| 3.7.1   | Descrizione                                | 13 |
| 3.7.2   | Ciclo di vita                              | 13 |
| 3.7.2.1 | Individuazione                             | 13 |
| 3.7.2.2 | Analisi problematiche                      | 13 |

|          |   |           |
|----------|---|-----------|
| 3.7.2.3  | Scelta soluzione migliore . . . . .                 | 14        |
| 3.7.2.4  | Reinserimento nella gestione di processo . . . . .  | 15        |
| <b>4</b> | <b>Processi Organizzativi</b>                       | <b>15</b> |
| 4.1      | Processi di Coordinamento . . . . .                 | 15        |
| 4.1.1    | Gestione Comunicazioni . . . . .                    | 15        |
| 4.1.1.1  | Comunicazioni Interne . . . . .                     | 15        |
| 4.1.1.2  | Comunicazioni Esterne . . . . .                     | 15        |
| 4.1.2    | Gestione Riunioni . . . . .                         | 15        |
| 4.1.2.1  | Riunioni Interne . . . . .                          | 15        |
| 4.1.2.2  | Riunioni Esterne . . . . .                          | 15        |
| 4.1.2.3  | Verbale di Riunione . . . . .                       | 15        |
| 4.2      | Ruoli di Progetto . . . . .                         | 15        |
| 4.2.1    | Responsabile di Progetto . . . . .                  | 16        |
| 4.2.2    | Amministratore di Progetto . . . . .                | 16        |
| 4.2.3    | Analista . . . . .                                  | 16        |
| 4.2.4    | Progettista . . . . .                               | 17        |
| 4.2.5    | Programmatore . . . . .                             | 17        |
| 4.2.6    | Verificatore . . . . .                              | 17        |
| 4.2.7    | Rotazione dei Ruoli . . . . .                       | 18        |
| 4.3      | Procedure . . . . .                                 | 18        |
| 4.3.1    | Gestione degli Strumenti di Versionamento . . . . . | 18        |
| 4.3.1.1  | Struttura Repository . . . . .                      | 18        |
| 4.3.1.2  | Tipi di files . . . . .                             | 18        |
| 4.3.1.3  | Norme delle Commit . . . . .                        | 18        |
| 4.3.2    | Gestione degli Strumenti di Coordinamento . . . . . | 18        |
| 4.3.2.1  | Tasks . . . . .                                     | 18        |
| 4.3.2.2  | Tickets . . . . .                                   | 18        |
| 4.4      | Strumenti . . . . .                                 | 18        |
| 4.4.1    | Sistema Operativo . . . . .                         | 18        |
| 4.4.2    | Versionamento e Issue Tracking . . . . .            | 19        |
| 4.4.2.1  | Git . . . . .                                       | 19        |
| 4.4.2.2  | GitHub . . . . .                                    | 19        |
| 4.4.3    | Comunicazione . . . . .                             | 19        |
| 4.4.3.1  | Telegram . . . . .                                  | 19        |
| 4.4.3.2  | Slack . . . . .                                     | 19        |
| 4.4.4    | Diagrammi di Gantt . . . . .                        | 19        |
| 4.4.5    | Diagrammi UML . . . . .                             | 20        |



|     |                                 |    |
|-----|---------------------------------|----|
| 4.5 | Formazione del Gruppo . . . . . | 20 |
| 5   | Changelog                       | 21 |



# 1 Introduzione

## 1.1 Scopo del Documento

## 1.2 Ambiguità e Glossario

## 1.3 Riferimenti

## 2 Processi Primari

### 2.1 Fornitura

In questa sezione del documento vengono trattate le norme che il team **Agents of S.W.E.** decide e si impegna a rispettare, con lo scopo di proporsi e divenire fornitori nei confronti dell'azienda proponente, *Zucchetti S.p.A.*, e dei committenti Prof. Tullio Vardanega e Prof. Riccardo Cardin nell'ambito della progettazione, sviluppo e consegna del prodotto "*Plugin per Grafana*".

### 2.2 Studio di fattibilità

### 2.3 Sviluppo

#### 2.3.1 Analisi dei requisiti

##### 2.3.1.1 Classificazione dei requisiti

##### 2.3.1.2 Classificazione dei casi d'uso

#### 2.3.2 Progettazione

#### 2.3.3 Codifica

Di seguito vengono definite delle norme che devono essere adottate dai Programatori per garantire una buona leggibilità e manutenibilità del codice. Le prime norme che seguiranno sono le più generali, da adottarsi per ogni linguaggio di programmazione adottato all'interno del progetto, in seguito quelle più specifiche per i linguaggi JavaScript<sub>G</sub>, HTML<sub>G</sub> e CSS<sub>G</sub>.

Ogni norma è caratterizzata da un paragrafo di appartenenza, da un titolo, una breve descrizione, e se il caso lo richiede, un esempio.

Il rispetto delle seguenti norme è fondamentale per garantire uno stile di codifica uniforme all'interno del progetto, oltre che per massimizzare la leggibilità e agevolare la manutenzione, la verifica<sub>G</sub> e la validazione<sub>G</sub>.

##### 2.3.3.1 Convenzioni per i nomi:

- I Programatori devono adottare come notazione per la definizione di cartelle, file, metodi, funzioni e variabili il CamelCase<sub>G</sub>.

Di seguito un esempio di corretta nomenclatura:

|                    |
|--------------------|
| 1 INSERIRE ESEMPIO |
|--------------------|

- Tutti i nomi devono essere **unici** ed **autoesplicativi**, ciò per evitare ambiguità e limitare la complessità .

### 2.3.3.2 Convenzioni per la documentazione:

- Tutti i nomi ed i commenti al codice vanno scritti in **inglese**;
- Nel codice è possibile utilizzare un commento con denominazione **TODO** in cui si vanno ad indicare compiti da svolgere;
- L'intestazione di ogni file deve essere la seguente:

```
1  /**
2  * File: nomeFile
3  * Type: fileType
4  * Creation date: gg/mm/yyyy
5  * Author: Name Surname
6  * Author e-mail: email@example.com
7  * Version: versionNumber
8  *
9  * Changelog:
10 * #entry || Author || Date || Description
11 */
```

- La versione del file nell'intestazione, deve rispettare la seguente formulazione: Y.K, dove Y rappresenta la versione principale, K la versione parziale della relativa versione principale.  
I numeri di versione del tipo Y.0, dalla 1.0, vengono considerate versioni stabili, e quindi versioni da testare per saggiarne la qualità .

**2.3.3.3 ECMAScript 6:** Seguendo le indicazioni presenti nella documentazione<sup>1</sup> dell'azienda fornitrice di *Grafana*, la piattaforma per cui si intende sviluppare il plugin, il team ha deciso di adottare come linguaggio di programmazione principale ECMAScript 6<sup>2</sup>.

ECMAScript 6 viene standardizzato da **ECMA<sub>G</sub>**<sup>3</sup> nel giugno 2015 con la sigla **ECMA-262**<sup>4</sup>.

Come stile di codifica si adottano le linee guida proposte da **Airbnb JavaScript**

<sup>1</sup><http://docs.grafana.org/plugins/developing/development/>

<sup>2</sup>Linguaggio divenuto standard ISO: ISO/IEC 16262:2011, e relativo aggiornamento ISO/IEC 22275:2018.

<sup>3</sup><http://www.ecma-international.org/>

<sup>4</sup><https://www.ecma-international.org/ecma-262/6.0/>



**Style Guide**<sup>5</sup>. Per la verifica dell'adesione a tali norme, i Programmatori devono utilizzare, come suggerito dalla documentazione proposta da *Grafana*, **ESLint**<sub>G</sub><sup>6</sup>. In particolare i Programmatori devono rispettare 5 linee guida proposte dalla documentazione ufficiale di *Grafana*:

1. Se una variabile non viene riutilizzata, deve essere dichiarata come **const**;
2. Utilizzare preferibilmente, per la definizione di variabili, la keyword **let**, anziché **var**;
3. Utilizzare il marcatore freccia (**=>**), in quanto non oscura il **this**:

```
1 testDatasource() {  
2   return this.getServerStatus()  
3   .then(status => {  
4     return this.doSomething(status);  
5   })  
6 }
```

Invece che:

```
1 testDatasource() {  
2   var self = this;  
3   return this.getServerStatus()  
4     .then(function(status) {  
5       return self.doSomething(status);  
6     })  
7 }
```

4. Utilizzare l'oggetto *Promise*:

```
1 metricFindQuery(query) {  
2   if (!query) {  
3     return Promise.resolve([]);  
4   }  
5 }
```

Invece che:

```
1 metricFindQuery(query) {  
2   if (!query) {  
3     return this.$q.when([]);  
4   }  
5 }
```

<sup>5</sup><https://github.com/airbnb/javascript>

<sup>6</sup><https://eslint.org/>

5. Se si utilizza *Lodash* è meglio essere consequenti, e preferire in ogni caso le funzioni per gli array native di ES6.

Verranno esaminate di seguito le norme in merito allo stile di codifica che i Programmatori dovranno adottare.

### Indentazione

**Norma 1** L'indentazione è da eseguirsi con tabulazione la cui larghezza sia impostata a due (2) spazi per ogni livello.

Di seguito un esempio da ritenersi corretto:

```
1 function() {  
2   ..let x = 2;  
3   ..if (x > 0)  
4     ....return true;  
5   ..else  
6     ....return false;  
7 }
```

Qualsiasi altro tipo di indentazione è da ritenersi scorretta.

**Norma 2** Dopo la graffa principale va inserito uno (1) spazio. Nel seguente modo:

```
1 function() { ... }
```

**Norma 3** Dopo la keyword di un dato statement (*if*, *while*, etc.) va inserito uno (1) spazio. Per un esempio corretto si veda la norma successiva.

**Norma 4** Prima dell'apertura della graffa negli statement di controllo va inserito uno (1) spazio. Nel seguente modo:

```
1 function() {  
2   if (condition) {  
3     ...  
4   }  
5   while (condition) {  
6     ...  
7   }  
8 }
```

**Norma 5** Negli statement di controllo (*if*, *while*, etc) le condizioni concatenate, o annidate, mediante operatori logici, che diventano eccessivamente lunghe NON vanno espresse in un'unica linea, bensì spezzate in più righe. Nel seguente modo:

```
1 function() {  
2     if (condition && condition) {  
3         ...  
4     }  
5  
6     if (  
7         veryLongCondition  
8         && longCondition  
9         && condition  
10    ) {  
11        doSomething();  
12    }  
13 }
```

**Norma 6** Dopo blocchi, o prima di un nuovo statement va lasciata una riga vuota.  
Nel seguente modo:

```
1 function1() {  
2     if (condition) {  
3         doSomething():  
4     }  
5  
6     return toReturn;  
7 }  
8  
9 function2(){  
10     ...  
11 }
```

**Norma 7** I blocchi di codice multi-riga devono essere contenuti all'interno di graffe.  
Blocchi costituiti da una singola riga non è necessario che sia contenuti tra graffe:  
nel caso non vengano utilizzate, la definizione deve essere *inline*, sulla stessa riga.  
Nel seguente modo:

```
1 if (condition) return true;  
2  
3 if (condtion) {  
4     return true;  
5 }
```

### Commenti al codice

Il codice va commentato nel seguente modo:

- "//" se il commento occupa una sola riga;
- "/\* ... \*/" se il commento occupa più righe.

Nel seguente modo:

```
1 // single line comment
2 if (condition) return true;
3
4 /**
5  * multi line comment, line 1
6  * multi line comment, line 2
7  */
8 if (condtion) {
9     return true;
10 }
```

## Variabili

**Norma 1** Fare riferimento alle norme 1 e 2, delle 5 linee guida enunciate sopra.

**Norma 2** Non utilizzare dichiarazioni multiple di variabili, dichiarare una variabile per riga.

Nel seguente modo:

```
1 // OK
2 var x = 1;
3 var y = 0;
4
5 // NO
6 var x = 1, y = 0;
```

## Nomi

**Norma 1** Oltre a quanto enunciato nel secondo punto del paragrafo §2.2.3.1, tutti i nomi di funzioni o variabili composti da una singola lettera, o che indichino temporaneità della variabile sono *vietati*: ogni nome deve essere significativo.

## Norma 2

1. I nomi delle variabili, funzioni ed istanze devono utilizzare il **CamelCase**;
2. I nomi delle classi deve lo stile **capWords**.

Nel seguente modo:



```
1 // OK
2 var thisIsAVariable;
3
4 function thisIsAFunction() { ... }
5
6 class ThisIsAClass() {
7     ...
8 }
9
10 // NO
11 var Variable;
12
13 function Function() { ... }
14
15 class myClass() {
16     ...
17 }
```

#### 2.3.3.4 HTML

## 3 Processi di supporto

### 3.1 Documentazione

#### 3.1.1 Descrizione

Questo capitolo descrive i dettagli su come deve essere redatta e verificata la documentazione durante il ciclo di vita del software. Le norme sono tassativamente valide per tutti i documenti formali.

#### 3.1.2 Ciclo di vita documentazione

Il ciclo di vita previsto della documentazione si può suddividere principalmente in tre processi:

- **Sviluppo:** è il processo di stesura, eseguita dal redattore, dove descrive il ticket<sub>G</sub> assegnato dal responsabile. Una volta terminata la fase di scrittura del documento, il redattore lo segnala al responsabile, che assegnerà a un verificatore il compito di analizzare il lavoro svolto;
- **Verifica:** è il processo eseguito dai verificatori designati dal responsabile, il loro compito è controllare che il redattore abbia scritto il documento nella norma e in maniera corretta grammaticalmente e strutturalmente;
- **Approvato:** è il processo conclusivo, in cui il verificatore ha terminato il suo compito di controllo e comunica al responsabile il termine del lavoro. Il responsabile procederà a confermare il documento e ad eseguire il rilascio.

#### 3.1.3 Template

Il gruppo ha deciso di strutturare un template L<sup>A</sup>T<sub>E</sub>X per dare uniformità a tutti i documenti. Il template facilita e velocizza la stesura, poiché i redattori devono concentrarsi solo ed esclusivamente al contenuto e non alla layout.

#### 3.1.4 Struttura documenti

Ogni documento segue una determinata struttura, predefinita e accordata dal gruppo:

##### 3.1.4.1 Prima pagina :



#### 3.1.4.2 Piè di pagina :

#### 3.1.4.3 Nomenclatura :

#### 3.1.4.4 Tabelle :

#### 3.1.4.5 Indice Sezioni :

#### 3.1.5 Documenti Correnti

Sono descritti brevemente i documenti formali da consegnare:

- Analisi dei Requisiti:
- Glossario:
- Norme di Progetto:
- Piano di Progetto:
- Piano di Qualifica:
- Studio di Fattibilità:

#### 3.1.6 Norme

##### 3.1.6.1 Struttura dei documenti

##### 3.1.6.2 Norme tipografiche

### 3.1.7 Struttura documentazione

### 3.1.8 Gestione termini Glossario

### 3.1.9 Ambiente

## 3.2 Qualità

### 3.2.1 Descrizione

### 3.2.2 Classificazione dei processi

### 3.2.3 Procedure

## 3.3 Versionamento

La necessità di più componenti del gruppo di cooperare su uno stesso documento, porta alla soluzione di utilizzare un sistema di versionamento distribuito.

### 3.3.1 Controllo di versione

Il sistema di versionamento, utilizzato in questa fase di **RR**, è **git**, con il supporto hosting di **GitHub**.

**3.3.1.1 Struttura del repository** La struttura del repository segue il workflow **gitflow** di **Driessen at nvie**, idealizzato attorno il concetto di **release** del prodotto. Questo produce un framework robusto attorno al quale si possono gestire progetti di grandi dimensioni. I due branch principali sono il **master** e in parallelo ad esso il **develop**. Il **master** viene considerato il branch *main*, dove il codice sorgente della *testa* riflette sempre lo stato di "*production-ready*", mentre il ramo di **develop** è considerato il branch principale dove vengono effettuate le ultime modifiche per il prossimo rilascio del prodotto.

**3.3.1.2 Processo di implementazione** L'implementazione dei documenti avviene tramite gli strumenti utilizzati nel paragrafo –INSERIRE PARAGRAFO PARTE DI FAVARO–

Quest'ultimi, in fase di compilazione producono dei file di poca rilevanza con estensioni come: *.log*, *.out*, *.idx*, *.aux*, *.gz*, *.aux*, *.toc*, i quali verranno ignorati come da configurazione, ma soprattutto file di più rilevanza come *.pdf*, *.tex*, i quali verranno versionati dal sistema di git preinstallato. Una volta creati/modificati i documenti, si procede con il `g` commit di essi. Il commit riporta un *cambiamento*



al file, con un messaggio allegato ad esso che ne descrive le modifiche apportate o un comando apposito per chiudere alcune task con tale commit. Dopo di che, il commit viene  $\hookrightarrow$  pushato nel branch appropriato, a seconda dei criteri descritti nel prossimo paragrafo.

### 3.3.1.3 Ciclo di vita dei branch

1. **Master:** branch *main* del repository, esso rappresenta lo stato di "*production-ready*" del prodotto. Questo branch ha una durata di vita quanto il repository stesso o infinita;
2. **Develop:** branch di sviluppo parallelo al **master** sul quale vengono aggiunte le feature provenienti appunto dai branch **feature**, e dal quale inizia il branch di **release**. Ha una durata di vita quanto il branch master;
3. **Release:** branch di preparazione per un nuovo rilascio o aggiornamento del prodotto. Utilizzato per risolvere piccoli errori e configurare le impostazioni di rilascio. Una volta rilasciato il prodotto, esso si riversa sul branch **master** e **develop**. Ha una durata breve in quanto il rilascio deve essere effettuato il prima possibile;
4. **Feature:** branch usato per sviluppare nuove feature per il prossimo rilascio a breve o lungo tempo. Il suo tempo di vita dura quanto lo sviluppo della nuova feature fintanto che non avviene il  $\hookrightarrow$  merge sul branch di **develop**;
5. **Hotfix:** branch molto simili a quelli di release, con l'obiettivo di risolvere immediatamente un bug del prodotto in produzione o release. Una volta risolto il bug, esso si riversa sui branch **master** e **develop**, aggiornandoli nel minor tempo possibile. Ha un tempo di vita breve, in quanto viene creato per la necessità di risolvere un problema sul prodotto rilasciato.

### 3.3.1.4 Rilascio di versione

## 3.3.2 Configurazione versionamento

### 3.3.2.1 Remoto

### 3.3.2.2 Locale



## 3.4 Gestione di progetto

### 3.4.1 Configurazione strumenti di organizzazione

#### 3.4.1.1 Inizializzazione

#### 3.4.1.2 Aggiunta tasks

#### 3.4.1.3 Aggiunta milestones

### 3.4.2 Ciclo di vita delle tasks

#### 3.4.2.1 Apertura

#### 3.4.2.2 Completamento

#### 3.4.2.3 Richiesta di revisione

#### 3.4.2.4 Completamento

## 3.5 Verifica

### 3.5.1 Descrizione

### 3.5.2 Analisti Statica

### 3.5.3 Verifica Diagrammi UML

### 3.5.4 Strumenti

## 3.6 Validazione

### 3.6.1 Descrizione

### 3.6.2 Procedure

## 3.7 Risoluzione problematiche

### 3.7.1 Descrizione

### 3.7.2 Ciclo di vita

#### 3.7.2.1 Individuazione

#### 3.7.2.2 Analisi problematiche



### 3.7.2.3 Scelta soluzione migliore

#### 3.7.2.4 Reinserimento nella gestione di processo

## 4 Processi Organizzativi

### 4.1 Processi di Coordinamento

#### 4.1.1 Gestione Comunicazioni

##### 4.1.1.1 Comunicazioni Interne

##### 4.1.1.2 Comunicazioni Esterne

#### 4.1.2 Gestione Riunioni

##### 4.1.2.1 Riunioni Interne

##### 4.1.2.2 Riunioni Esterne

##### 4.1.2.3 Verbale di Riunione

### 4.2 Ruoli di Progetto

Nell'ottica di un lavoro ben organizzato e collaborativo tra i membri del gruppo, ad ogni componente, in ogni momento, è attribuito un ruolo per un periodo di tempo limitato.

Questi ruoli, che corrispondono ad una figura aziendale ben precisa, sono:

- **Responsabile di Progetto;**
- **Amministratore di Progetto;**
- **Analista;**
- **Progettista;**
- **Programmatore;**
- **Verificatore.**

#### 4.2.1 Responsabile di Progetto

Detto anche "*Project Manager*", è il rappresentante del progetto<sub>G</sub>, agli occhi sia del committente che del fornitore. Egli risulta dunque essere, in primo luogo, il responsabile ultimo dei risultati del proprio gruppo. Figura di grande responsabilità, partecipa al progetto<sub>G</sub> per tutta la sua durata, ha il compito di prendere decisioni e approvare scelte collettive.

Nello specifico egli ha la responsabilità di:

- Coordinare le attività del gruppo, attraverso la gestione delle risorse umane;
- Approvare i documenti redatti, e verificati, dai membri del gruppo;
- Elaborare piani e scadenze, monitorando i progressi nell'avanzamento del progetto<sub>G</sub> ;
- Redigere l'organigramma del gruppo e il *Piano di Progetto<sub>G</sub>* .

#### 4.2.2 Amministratore di Progetto

L'amministratore è la figura chiave per quanto concerne la produttività. Egli ha infatti come primaria responsabilità il garantire l'efficienza<sub>G</sub> del gruppo, fornendo strumenti utili e occupandosi dell'operatività delle risorse. Ha dunque il compito di gestire l'ambiente lavorativo.

Tra le sue responsabilità specifiche figurano:

- Redigere documenti che normano l'attività lavorativa, e la loro verifica<sub>G</sub> ;
- Redigere le *Norme di Progetto<sub>G</sub>* ;
- Scegliere ed amministrare gli strumenti di versionamento<sub>G</sub> ;
- Ricerare strumenti che possano agevolare il lavoro del gruppo;
- Attuare piani e procedure di gestione della qualità<sub>G</sub> .

#### 4.2.3 Analista

L'analista deve essere dotato di un'ottima conoscenza riguardo al dominio del problema. Egli ha infatti il compito di analizzare tale dominio e comprenderlo appieno, affinché possa avvenire una corretta progettazione<sub>G</sub> .

Ha il compito di:

- Comprendere al meglio il problema, per poi poterlo esporre in modo chiaro attraverso specifici requisiti<sub>G</sub> ;
- Redarre lo *Studio di Fattibilità* e l'*Analisi dei Requisiti*<sub>G</sub> .

#### 4.2.4 Progettista

Il progettista è responsabile delle attività di progettazione<sub>G</sub> attraverso la gestione di aspetti tecnici del progetto<sub>G</sub> .

Più nello specifico si occupa di:

- Definire l'Architettura<sub>G</sub> del prodotto<sub>G</sub> , applicando quanto più possibile norme di *best practice*<sub>G</sub> , prestando attenzione alla manutenibilità del prodotto<sub>G</sub> ;
- Suddividere il problema, e di conseguenza il sistema, in parti di complessità trattabile.

#### 4.2.5 Programmatore

Il programmatore si occupa delle attività di codifica, le quali portano alla realizzazione effettiva del prodotto<sub>G</sub> .

Egli ha dunque il compito di:

- Implementare l'architettura definita dal *Progettista*, prestando attenzione a scrivere codice coerente con ciò che è stato stabilito nelle norme di qualifica;
- Produrre codice documentato e manutenibile;
- Realizzare le componenti necessarie per la verifica<sub>G</sub> e la validazione<sub>G</sub> del codice;
- Redarre il *Manuale Utente*.

#### 4.2.6 Verificatore

Il verificatore, figura presente per l'intera durata del progetto<sub>G</sub> , è responsabile delle attività di verifica<sub>G</sub> .

Nello specifico egli:

- Verifica l'applicazione ed il rispetto delle *Norme di Progetto*<sub>G</sub> ;
- Segnala al *Responsabile di Progetto* l'emergere di eventuali discordanze tra quanto presentato nel *Piano di Progetto*<sub>G</sub> e quanto effettivamente realizzato;
- Ha il compito di redarre il *Piano di Qualifica*.

#### **4.2.7 Rotazione dei Ruoli**

Come da istruzioni ogni membro del gruppo dovrà ricoprire, per un periodo di tempo limitato, ciascun ruolo, nel rispetto delle seguenti regole:

- Ciascun membro dovrà svolgere esclusivamente le attività proprie del ruolo a lui assegnato;
- Al fine di evitare conflitti di interesse nessun membro potrà ricoprire un ruolo che preveda la verifica di quanto da lui svolto, nell'immediato passato;
- Vista l'ampia differenza di compiti e mansioni tra i vari ruoli, e al fine di valorizzare l'attività collaborativa all'interno del gruppo, ogni componente che abbia ricoperto in precedenza un ruolo ora destinato a qualcun altro dovrà fornire supporto al compagno in caso di necessità, fornendogli consigli e, se possibile, affiancandolo in situazioni critiche.

### **4.3 Procedure**

#### **4.3.1 Gestione degli Strumenti di Versionamento**

##### **4.3.1.1 Struttura Repository**

##### **4.3.1.2 Tipi di files**

##### **4.3.1.3 Norme delle Commit**

#### **4.3.2 Gestione degli Strumenti di Coordinamento**

##### **4.3.2.1 Tasks**

##### **4.3.2.2 Tickets**

### **4.4 Strumenti**

#### **4.4.1 Sistema Operativo**

I sistemi operativi utilizzati dai membri del gruppo sono i seguenti:

- Windows 10 x64;
- Mac OS 10.14.1;
- Majaro Linux - 4.14.85-1 LTS.

#### 4.4.2 Versionamento e Issue Tracking

##### 4.4.2.1 Git

*Git* è il sistema di versionamento<sub>G</sub> scelto dal gruppo. E' un sistema open source creato da Linus Torvalds 2005. Presenta un'interfaccia a riga di comando, tuttavia esistono svariati *tools<sub>G</sub>* che ne forniscono una GUI.

##### 4.4.2.2 GitHub

*Github* è un *Respository Manager<sub>G</sub>* che usa *Git* come sistema di versioning, offre inoltre servizi di *issue tracking<sub>G</sub>* .

#### 4.4.3 Comunicazione

##### 4.4.3.1 Telegram

Telegram è una delle maggiori e più note applicazioni di messaggistica istantanea *cross platform*, utilizzabile contemporaneamente su più dispositivi. Oltre alla semplice messaggistica offre servizi quali lo scambio di files, la creazione di gruppi e le chiamate vocali.

##### 4.4.3.2 Slack

Slack è un'applicazione di messaggistica istantanea specializzata nella comunicazione interna tra membri di un gruppo di lavoro. L'applicazione è organizzata in *workspace*, a loro volta suddivisi in canali, i quali consentono di catalogare le conversazione sulla base dell'argomento trattato. Questa struttura, studiata appositamente per l'ambito lavorativo, è stata giudicata come positiva e vantaggiosa dal gruppo, visto che consente di mantenere chat ordinate e monotematiche.

Nonostante preveda anche abbonamenti a pagamento le funzionalità base di Slack sono gratuite, inoltre, come Telegram, risulta essere un'applicazione *cross platform<sub>G</sub>* .

#### 4.4.4 Diagrammi di Gantt

Lo strumento scelto dal gruppo per la realizzazione dei diagrammi di Gantt<sub>G</sub> è "Gantt Project". Le motivazioni che hanno portato a questa scelta sono molteplici, tra queste spiccano il fatto che sia uno strumento gratuito, *open-source<sub>G</sub>* , e *cross platform*. L'elevata accessibilità è stata infatti giudicata come una caratteristica di primaria importanza, considerando i differenti sistemi operativi utilizzati dai componenti del gruppo.



#### 4.4.5 Diagrammi UML

Lo strumento scelto dal gruppo per la realizzazione dei diagrammi UML è "Umbrello", un software *open-source*<sub>G</sub> per il disegno di diagrammi UML 2.0, creato da KDE Team. Anche in questo caso è stata tenuta in grande considerazione l'accessibilità dello strumento. Il software in questione risulta infatti gratuito e disponibile sia su Windows, sia su Mac OS tramite porting, sebbene sia nativo Unix.

Umbrello risulta essere sia *User Friendly*, con un'interfaccia utente chiara e consistente, sia abbastanza potente da essere utilizzato anche per le industrie. Questo, unito alle dimensioni relativamente ridotte del software, e al fatto che non sia richiesta alcuna licenza per l'installazione, hanno fatto propendere il gruppo per questo programma, a discapito di alternative, come Papyrus o Astah, comunque considerate valide.

#### 4.5 Formazione del Gruppo

La formazione dei componenti del gruppo **Agents of S.W.E.** è da considerarsi individuale. Ogni membro del team è infatti tenuto a documentarsi autonomamente riguardo le tecnologie coinvolte nello sviluppo del progetto<sub>G</sub>. Tuttavia, nell'ottica di un ambiente di lavoro sano e collaborativo, nel caso in cui fosse necessario, è compito degli *Amministratori* mettere a disposizione di chi ne avesse bisogno risorse basilari, al fine di agevolare la formazione dei restanti componenti del gruppo.

## 5 Changelog

| Versione | Data       | Autore         | Ruolo          | Descrizione                            |
|----------|------------|----------------|----------------|--|
| 0.0.1    | 2018-11-23 | Luca Violato   | Amministratore | Strutturazione del Documento           |
| 0.0.2    | 2018-11-23 | Marco Chiese   | Verificatore   | Prima stesura §2                       |
| 0.0.3    | 2018-12-01 | Luca Violato   | Amministratore | Strutturazione §4, stesura §4.2 e §4.5 |
| 0.0.4    | 2018-12-03 | Bogdan Stanciu | Responsabile   | Stesura §3                             |
| 0.0.5    | 2018-12-04 | Luca Violato   | Amministratore | Stesura §4.4                           |

**Tabella 1:** Changelog del documento