CLIPS

Norme di Progetto v4.0.0

Sommario
Questo documento definisce le norme stabilite dal gruppo Beacon Strips per la realizzazione del *progetto*$_g$ CLIPS

Nome del documento	Norme di Progetto
Versione	4.0.0
Data di redazione	2016-09-04
Redazione	Luca Soldera
Verifica	Viviana Alessio
Approvazione	Andrea Grendene
Uso	Interno
Lista di distribuzione	prof. Tullio Vardanega prof. Riccardo Cardin Miriade SpA

Diario delle modifiche

Versione	Riepilogo	Autore	Ruolo	Data
4.0.0	Approvazione documento	Enrico Grendene	Responsabile	2016-09-04
3.1.0	Verifica documento	Viviana Alessio	Verificatore	2016-09-04
3.0.2	Modificata sezione Analisi dinamica per togliere ripetizioni dal PdQ	LucaSoldera	Amministratore	2016-09-02
3.0.1	Aggiunta sezione Strumenti per i processi di verifica	LucaSoldera	Amministratore	2016-09-02
3.0.0	Approvazione documento	Tommaso Panozzo	Responsabile	2016-08-17
2.4.0	Verifica documento	Viviana Alessio	Verificatore	2016-07-10
2.3.0	Ampliata sezione Codifica con sottosezione Norme stilistiche	Luca Soldera	Amministratore	2016-07-08
2.2.0	Verifica documento	Matteo Franco	Verificatore	2016-07-06
2.1.0	Corretta e ampliata sezione Attività nel processo di sviluppo	Luca Soldera	Amministratore	2016-07-05
2.0.0	Approvazione documento	Matteo Franco	Responsabile	2016-06-10
1.2.0	Verifica documento	Luca Soldera	Verificatore	2016-05-10
1.1.0	Aggiunta sezione Codifica	Enrico Bellio	Amministratore	2016-05-08
1.0.0	Approvazione documento	Viviana Alessio	Responsabile	2016-03-15
0.6.0	Verifica finale	Enrico Bellio	Amministratore	2016-03-12
0.5.4	Correzione finale	Luca Soldera	Amministratore	2016-03-11
0.5.3	Verifica documento	Enrico Bellio	Verificatore	2016-03-11
0.5.2	Correzione sezione *Repository*$_{g}$ relativa al Processo di gestione	Luca Soldera	Amministratore	2016-03-09
0.5.1	Correzione sezione Norme relativa al Processo di documentazione	Matteo Franco	Amministratore	2016-03-09
0.6.0	Stesura sezioni Procedure e Norme relative al Processo di gestione	Luca Soldera	Amministratore	2016-03-08

Versione	Riepilogo	Autore	Ruolo	Data
0.5.0	Stesura sezioni Attività e Strumenti relative al Processo di gestione	Matteo Franco	Amministratore	2016-03-07
0.4.0	Stesura sezione Processi primari	Luca Soldera	Amministratore	2016-03-06
0.3.0	Completamento stesura sezione Processo di documentazione	Matteo Franco	Amministratore	2016-03-06
0.2.1	Stesura sezione Processo di verifica	Luca Soldera	Amministratore	2016-03-04
0.2.0	Inizio stesura sezione Processo di documentazione	Matteo Franco	Amministratore	2016-03-04
0.1.0	Creazione scheletro documento e stesura sezione Introduzione	Matteo Franco	Amministratore	2016-03-03

Indice

1 Introduzione

1.1 Scopo del documento

Questo documento intende definire le norme da rispettare all'interno del gruppo Beacon Strips durante lo sviluppo del *progetto*$_g$ CLIPS.
Tutti i membri del team sono tenuti a visionare il documento e a rispettare le norme. 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.

1.2 Scopo del *prodotto*$_g$

Il prodotto finale consisterà di un'applicazione mobile che, interagendo con dei beacons sparsi nell'area designata, guiderà l'utente attraverso un percorso. L'utente potrà completare il percorso superando tutte le prove che gli si presenteranno nelle diverse tappe. Le prove potranno essere degli indovinelli o dei semplici giochi inerenti all'area in cui si svolge il percorso.

1.3 Glossario

Al fine di evitare ogni ambiguità nel linguaggio e massimizzare la comprensione dei documenti, i termini tecnici, gli acronimi e le abbreviazioni che necessitano di definizione sono riportati nel documento "*Glossario v3.0.0*".

Inoltre ogni occorrenza di un vocabolo presente nel Glossario sarà posta in corsivo e seguita da una 'g' minuscola a pedice (p.es. *Glossario*$_g$).

1.4 Riferimenti

1.4.1 Riferimenti informativi

- ***ISO*$_g$ 8601**: http://www.iso.org/iso/home/standards/iso8601.htm;
- ***GitHub*$_g$**: http://github.com/.

2 Processi primari

2.1 Processo di sviluppo

2.1.1 Attività

2.1.1.1 Analisi dei requisiti

2.1.1.1.1 Studio di Fattibilità e Analisi dei Rischi

Prima dell'attività di analisi, sarà necessario effettuare uno studio di fattibilità per analizzare i vari capitolati, evidenziando di ciascuno gli aspetti positivi e negativi, in maniera particolare per il capitolato scelto. Questa attività porterà alla produzione del documento "*Studio di Fattibilità v2.0.0*". Lo studio di fattibilità sarà il primo documento dell'analisi dei requisiti che analizzerà i seguenti punti per ogni *capitolato*$_{\text{g}}$, con maggior interesse per quello scelto dal team:

- **Scopo del *progetto*$_{\text{g}}$**: analisi delle richieste del *capitolato*$_{\text{g}}$;
- **Studio del dominio**: valutazione delle tecnologie e conoscenze richieste rispetto l'attuale livello del team;
- **Analisi dei rischi**: ricerca dei rischi e delle criticità per ciascun *capitolato*$_{\text{g}}$.

2.1.1.1.2 Analisi dei requisiti

Una volta terminato lo studio di fattibilità si procederà all'analisi dei requisiti.
Questa attività consiste nell'identificare i requisiti e i casi d'uso a partire dalle informazioni raccolte tramite lo studio del *capitolato*$_{\text{g}}$ e riunioni esterne con il *proponente*$_{\text{g}}$. I requisiti prodotti dovranno essere atomici e tracciabili; i casi d'uso dovranno coprire tutte le possibile interazioni dell'utente.
Per rendere più precisa e veloce la stesura dei requisiti e fornire un tracciamento di essi verrà utilizzato il *software*$_{\text{g}}$ *Trender*$_{\text{g}}$.
L'attività di analisi concluderà con la redazione del documento "*Analisi dei Requisiti v4.0.0*".

2.1.1.2 Progettazione

L'attività di progettazione sarà caratterizzata dal descrivere l'architettura ad alto livello dell'applicazione e progettare opportuni test d'integrazione. Questi compiti saranno svolti dai Progettisti.
Nello specifico si dovranno identificare:

- l'architettura da realizzare;
- eventuali *design pattern*$_{\text{g}}$ da utilizzare: i progettisti dovranno inoltre fornire una descrizione e un'immagine esplicativa dei design pattern utilizzati per realizzare l'applicazione;
- i package, per i quali sarà necessario realizzare i corrispondenti diagrammi;
- le classi, per le quali sarà necessario realizzare i corrispondenti diagrammi;
- le attività svolte dall'utente; per meglio identificare le attività i progettisti dovranno analizzare i casi d'uso precedentemente identificati.

Una volta identificati tutti i componenti necessari, si procederà alla produzione del documento "*Specifica Tecnica v3.0.0*".

2.1.1.2.1 Tracciamento componenti

Affinchè tutti i requisiti vengano soddisfatti, verrà utilizzata l'applicazione *Trender*$_{\text{g}}$ per tenere traccia delle varie componenti e dei requisiti che soddisfano.

2.1.1.2.2 Test di integrazione

Verranno progettate delle classi il cui scopo sarà quello di testare la correttezza dei componenti dell'applicazione.

2.1.1.3 Codifica

Una volta ultimata l'attività di progettazione ci si occuperà di approfondire il livello di dettaglio, identificando anche metodi e campi dati da utilizzare nelle classi. Questa analisi approfondita porterà alla produzione della "*Definizione di Prodotto v2.0.0*".
Successivamente ci si occuperà di codificare tutte le scelte effettuate per poter realizzare l'applicazione richiesta.

2.1.2 Norme

2.1.2.1 Classificazione dei requisiti

I requisiti saranno rappresentati secondo la seguente codifica:

R[importanza][tipo][identificativo]

- **Importanza**:indica se il requisito è:
  1. **0**: obbligatorio;
  2. **1**: desiderabile;
  3. **2**: opzionale.
- **Tipo**: indica se è di tipo:
  1. **F**: funzionale;
  2. **Q**: di qualità;
  3. **P**: prestazionale;
  4. **V**: di vincolo.
- **Identificativo**: è il codice univoco e gerarchico che automaticamente il *software*$_{\text{g}}$ assegna al requisito (esempio: 4.2.1);
- **Descrizione**: una breve descrizione del requisito;
- **Fonte**: la fonte da cui deriva il requisito.

2.1.2.2 Classificazione dei casi d'uso

I casi d'uso saranno rappresentati secondo la seguente codifica:

UC[identificativo]

- **Identificativo**: codice univoco e gerarchico che automaticamente il *software*$_{\text{g}}$ assegna al caso d'uso (esempio: 3.4.1);

inoltre i casi d'uso saranno caratterizzati da:

- **Tipo**: se non specificato è di tipo standard altrimenti viene scelto fra:
  1. **estensione**;
  2. **inclusione**;
  3. **generalizzazione**.
- **titolo** breve del caso d'uso;
- **descrizione** del caso d'uso;
- **precondizione** del caso d'uso;
- **postcondizione** del caso d'uso
- **scenario principale** degli eventi;
- **scenario secondario** eventuale;
- **attori**: lista degli attori coinvolti;

2.1.3 Strumenti

2.1.3.1 *Trender*$_{\mathbf{g}}$

Per velocizzare e automatizzare la stesura dei requisiti e dei casi d'uso è stato utilizzato il *software*$_{\text{g}}$ ***Trender*$_{\mathbf{g}}$** sviluppato da Simone Campagna del gruppo InfiniTech dell'anno 2014/2015. Il *software*$_{\text{g}}$ permette di tracciare i requisiti e i casi d'uso assegnando automaticamente un codice univoco che rispetti la sintassi desiderata.

2.1.4 Codifica

2.1.4.1 Norme tipografiche

Tutto il codice prodotto dovrà seguire le seguenti regole:

- i nomi di variabili, metodi, funzioni e classi dovranno essere scritti in inglese;
- i nomi delle classi dovranno rispettare le regole del *CamelCase*$_{\text{g}}$;
- i nomi di variabili, metodi e funzioni dovranno avere la prima lettera minuscola e l'iniziale di ogni altra parola contenuta nel nome in maiuscolo;
- i commenti dovranno essere scritti in italiano;
- non vanno usate lettere accentate.

2.1.4.2 Norme stilistiche

2.1.4.2.1 Intestazione

Ogni file di codice dovrà contenere la seguente intestazione:

```
/**
* @file [nome del file]
* @date [data di creazione (gg/mm/aaaa)]
* @version [versione]
* @author [autore]
*
* Descrizione del file.
*/
```

```
        package nomePackage;
        public class NomeClasse {
                ...
        }
}
```

I costrutti e le classi devono essere scritti nel seguente formato:

```
        costrutto(condizione) {

        }

        tipoRitorno function () {
                istruzioni
        }
```

Nel caso il metodo sia get/set va utilizzato invece il seguente formato:

```
        tipoRitorno getVariable () { return variable; }
```

Le parentesi graffe vanno usate sempre, anche nel caso sia presente una sola istruzione nel costrutto.

2.1.4.3 Stile generale

Le seguenti regole vanno rispettati in tutti i file di codice:

- i file contenenti codice o documentazione dovranno rispettare la codifica UTF-8 senza BOM_g e verrà utilizzato il carattere `LF (U+000A)` per andare a capo;
- la tabulazione dev'essere di 3 spazi;
- è necessario commentare il codice nel caso vengano implementate funzionalità poco note, al fine di rendere più comprensibile il codice prodotto.

3 Processi di supporto

3.1 Processo di documentazione

3.1.1 Procedure

3.1.1.1 Ciclo di vita

Ogni documento può trovarsi in tre fasi differenti:

- **In lavorazione**: un documento è in lavorazione quando vengono aggiunti, modificati o rimossi dei contenuti;
- **Da verificare**: quando un documento è completo dovrà essere preso in consegna dai verificatori, che si occuperanno di rilevare e/o correggere errori sintattici e semantici;
- **Approvato**: un documento, ultimata la verifica, deve essere approvato dal Responsabile. L'approvazione determina lo stato finale della versione del documento.

Le varie fasi vengono scandite dal sistema di ticketing (vedi sezione 4.1.1.3).

3.1.2 Norme

3.1.2.1 *Template*$_{\text{g}}$

È presente una cartella nel *repository*$_{\text{g}}$ in `/documenti/template`.
La cartella contiene i seguenti files:

- *Comandi.sty*: contiene i comandi personalizzati (ad es. `\gl`);
- *Riferimenti.sty*: all'interno del file sono presenti vari comandi da utilizzare come scorciatoie (ad es. `\RES`);
- *Stile.sty*: contiene gli stili grafici da applicare al *template*$_{\text{g}}$;
- *TemplateDoc.tex*: è il *template*$_{\text{g}}$ da utilizzare per realizzare i documenti. Ogni documento dovrà essere realizzato a partire da questo file;
- *TemplateVerbale.tex*: questo *template*$_{\text{g}}$ va impiegato per realizzare i verbali. Ogni verbale dovrà essere realizzato a partire da questo file;
- *Glossario.sty*: questo file serve per impostare il *Glossario*, che avrà una configurazione diversa dal *template*$_{\text{g}}$ standard.

È presente inoltre una cartella *img* dove si trova il logo del team e dove potranno essere caricate tutte le altre immagini necessarie al *template*$_{\text{g}}$.

3.1.2.2 Norme tipografiche

In questa sezione vengono definite le norme riguardanti l'ortografia e la tipografia, al fine di avere uno stile uniforme per tutti i documenti prodotti.

3.1.2.2.1 Stile del testo

- **Grassetto**: il grassetto va utilizzato nei seguenti casi:
  - titoli;
  - elenchi puntati: può essere utilizzato il grassetto nel caso sia necessario evidenziare il concetto da sviluppare;
  - altri casi: per evidenziare parole chiave o contenuti importanti.
- **Corsivo**: lo stile corsivo si applica a:
  - documenti;
  - abbreviazioni.
- *LaTeX*$_{\text{g}}$: ogni occorrenza di *LaTeX*$_{\text{g}}$ va scritta con il comando `\LaTeX`
- Maiuscolo: è possibile utilizzare lo stile maiuscolo **solo** per gli acronimi;
- *Monospace*$_{\text{g}}$: le porzioni di testo scritte in *monospace*$_{\text{g}}$ definiscono:
  - frammenti di codice;
  - comandi;
  - URL.
- Glossario: le parole che hanno un riferimento nel glossario sono in corsivo e hanno una 'g' a pedice.

3.1.2.2.2 Punteggiatura

- **Spaziature**: ogni simbolo di punteggiatura deve essere seguito da uno spazio, e mai preceduto;
- **Parentesi**: le parentesi costituiscono un'eccezione riguardo le spaziature, in quanto devono essere precedute da uno spazio, ma non seguite;
- **Virgolette singole**: le virgolette singole indicano un singolo carattere;
- **Virgolette doppie**: le virgolette doppie vanno utilizzate per indicare citazioni e documenti.

3.1.2.2.3 Composizione del testo

- **Elenchi puntati**: ogni punto dell'elenco termina con un punto e virgola, ad eccezione dell'ultimo che deve terminare con un punto fermo.
  Ogni punto inizia con la minuscola, tranne nel caso in cui necessiti di una spiegazione: allora si utilizzerà la maiuscola;
- **Nota a piè di pagina**: ogni nota inizia con l'iniziale della prima parola maiuscola e termina con il punto. Non ci sono spaziature tra il numero della nota e il testo.

3.1.2.2.4 Formati

- **Date**: per le date va utilizzata la notazione definita dallo standard ISO_{g} 8601:2004:

  $$AAAA - MM - GG$$

  dove:
  - AAAA: rappresenta l'anno utilizzando quattro cifre;
  - MM: rappresenta il mese utilizzando due cifre;
  - GG: rappresenta il giorno utilizzando quattro cifre.
- **Orari**: gli orari rappresentati dovranno essere in linea con lo standard ISO_{g} 8601:2004:

  $$HH:MM$$

  dove:
  - HH: rappresenta l'ora utilizzando due cifre;
  - MM: rappresenta i minuti utilizzando due cifre.
- **Indirizzi**: per gli indirizzi verrà utilizzata la rappresentazione:

  Destinatario o luogo - Via e numero, CAP, Comune(Provincia)

  Nel caso l'indirizzo si riferisca ad uno Stato estero sarà necessario aggiungere il nome dello Stato dopo la provincia.
- **Sigle**: le sigle dei documenti vanno utilizzate solo nei diagrammi o nelle tabelle, con lo scopo di risparmiare spazio. Tali sigle sono:
  - **AdR** per "*Analisi dei Requisiti v4.0.0*";
  - **Gl** per "*Glossario v3.0.0*";
  - **NdP** per "*Norme di Progetto v4.0.0*";
  - **PdP** per "*Piano di Progetto v4.0.0*";

  - **PdQ** per "*Piano di Qualifica v4.0.0*";
  - **SdF** per "*Studio di Fattibilità v2.0.0*";
  - **ST** per "*Specifica Tecnica v3.0.0*".
- **Ruoli di *progetto*$_{\mathbf{g}}$**: per indicare i vari ruoli vanno utilizzati i seguenti comandi:
  - `\AM` per Amministratore;
  - `\AN` per Analista;
  - `\PR` per Programmatore;
  - `\PRJ` per Progettista;
  - `\RES` per Responsabile;
  - `\VER` per Verificatore.
- **Nomi propri**: per indicare nomi propri va utilizzata la forma *Nome Cognome*;
- **Nome del *proponente*$_{\mathbf{g}}$**: tramite il comando `\PROPONENTE` ci si riferisce al *proponente*$_{\text{g}}$ "Miriade SpA";
- **Nome del *progetto*$_{\mathbf{g}}$**: con `\PROGETTO` verrà citato il nome del *progetto*$_{\text{g}}$, ovvero CLIPS;
- **Nome del *committente*$_{\mathbf{g}}$**: `\COMMITTENTE` viene impiegato per riferirsi al "Prof. Tullio Vardanega".

3.1.2.3 Componenti grafiche

3.1.2.3.1 Immagini

Il formato preferito per le immagini è SVG (*Scalable Vector Graphics*$_{\text{g}}$), in quanto è possibile garantire una qualità maggiore e sono ridimensionabili senza perdere qualità. Nel caso non sia possibile integrare questo formato, si raccomanda l'utilizzo di immagini in formato PNG (*Portable Network Graphics*$_{\text{g}}$).

3.1.2.3.2 Tabelle

Le tabelle devono essere accompagnate da una didascalia e da un numero *incrementale*$_{\text{g}}$ per garantirne la tracciabilità.

3.1.2.4 Struttura del documento

3.1.2.4.1 Frontespizio

La prima pagina di ogni documento dovrà contenere le seguenti informazioni:

- nome del gruppo;
- nome del *progetto*$_{\text{g}}$;
- nome del documento e la sua 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;
- destinazione d'uso del documento;
- lista di distribuzione del documento.

3.1.2.4.2 Diario delle modifiche

La seconda pagina di ogni documento dovrà contenere il diario delle modifiche.
Il diario consiste in una tabella ordinata in modo decrescente secondo la data di modifica e, conseguentemente, al numero di versione. È particolarmente utile per tenere traccia delle varie modifiche effettuate nel documento.
Ogni riga del diario conterrà:

- numero di versione;
- breve riepilogo delle modifiche effettuate;
- autore delle modifiche;
- ruolo ricoperto dall'autore;
- data di modifica.

3.1.2.4.3 Indici

In ogni documento si trova un indice delle sezioni, il quale fornisce una visione macroscopica della struttura del documento. Nel caso comparissero tabelle e figure nel documento, potranno essere presenti i rispettivi indici, che dovranno apparire nel seguente ordine: indice delle sezioni, indice delle tabelle ed indice delle figure.

3.1.2.4.4 Intestazione e piè di pagina

L'intestazione di ogni pagina contiene i seguenti elementi:

- numero della sezione;
- titolo della sezione.

A piè di pagina invece si trovano:

- nome del documento e numero di versione, allineato a sinistra;
- nome del team, allineato al centro;
- pagina X di Y, dove X è la pagina corrente e Y è il numero di pagine totali, allineato a destra.

3.1.2.5 Verbali

I verbali verranno utilizzati per tenere traccia di informazioni importanti emerse durante alcune riunioni interne ed esterne.
Nella cartella `template` è presente un *template*$_{\text{g}}$ da utilizzare per redigere i verbali e sono presenti le seguenti sezioni:

- **Informazioni**: vanno indicate **sempre** le seguenti informazioni:
  - luogo;
  - data;
  - ora;

  - durata;
  - partecipanti.
- **Premessa**: la premessa consiste in una breve descrizione del motivo della riunione;
- **Ordine del giorno**: in questa sezione vanno indicati i vari punti da trattare durante la riunione;
- **Verbale**: nella sezione *Verbale* verranno trascritte tutte le informazioni rilevanti trattate durante la riunione.

3.1.2.6 Versionamento

I documenti prodotti sono soggetti al seguente versionamento attraverso la seguente codifica:

vX.Y.Z

dove:

- **X**: indica il numero crescente di uscite formali del documento;
- **Y**: indica il numero crescente di modifiche sostanziali al documento quali stesura, verifica, approvazione;
- **Z**: indica il numero crescente di modifiche minori effettuate sul documento.

I documenti vanno citati sempre con una versione specifica del documento, utilizzando i comandi:

- `\ARdoc` per il documento "*Analisi dei Requisiti v4.0.0*";
- `\Gldoc` per il documento "*Glossario v3.0.0*";
- `\NPdoc` per il documento "*Norme di Progetto v4.0.0*";
- `\PPdoc` per il documento "*Piano di Progetto v4.0.0*";
- `\PQdoc` per il documento "*Piano di Qualifica v4.0.0*";
- `\SFdoc` per il documento "*Studio di Fattibilità v2.0.0*";
- `\STdoc` per il documento "*Specifica Tecnica v3.0.0*".

Il formato da utilizzare per la creazione di file è:

`NomeDocumento_vX.Y.Z.pdf`

3.1.2.7 Strumenti

3.1.2.7.1 LaTeX$_g$

Per redigere la documentazione si è scelto di utilizzare il linguaggio di markup LaTeX$_g$, in quanto:

- permette di produrre documenti di alta qualità rispetto ai word processor tradizionali;
- rende possibile creare dei *template*$_g$ comuni per i documenti;
- permette una separazione tra contenuto e formattazione;
- è estendibile ed altamente personalizzabile tramite l'utilizzo di pacchetti specifici;
- è *multipiattaforma*$_g$, essendo un file sorgente di LaTeX$_g$ una semplice codifica *ASCII*$_g$;

Come editor di file *LaTeX*$_g$ è consigliato l'uso di *TeXstudio*, anch'esso *multipiattaforma*$_g$.

3.1.2.7.2 Correttore ortografico

In TeXstudio è presente un correttore ortografico automatico che permette la correzione in tempo reale. I file da utilizzare sono presenti nella cartella condivisa in *Google Drive*$_g$ all'indirizzo `/dizionari`.

3.1.2.7.3 Script

Su Github è presente una cartella `/script`$_g$ dove sono presenti degli *script*$_g$ utilizzabili dal gruppo per semplificare alcune operazioni.

3.2 Processo di verifica

3.2.1 Attività

3.2.1.1 Verifica di un *ticket*$_g$

Ogni qual volta un *ticket*$_g$ sarà contrassegnato come **Completato** il Responsabile procederà nel seguente modo:

- crea un *ticket*$_g$ di verifica;
- lo assegna ad un **verificatore**;
- non appena il *ticket*$_g$ viene contrassegnato come **Completato** imposta il tag del *ticket*$_g$ oggetto di verifica a **Verificato**.

3.2.1.2 Analisi statica

L'analisi statica è una tecnica di verifica applicabile sia ai documenti che al codice. Verrà effettuata per tutta la fase di sviluppo del *progetto*$_g$ e serve per trovare anomalie, che verranno trattate come descritto nelle "*Norme di Progetto v4.0.0*". Può essere applicata nei due modi seguenti.

3.2.1.2.1 Walkthrough

Consiste in una lettura del documento/codice cercando anomalie ed errori ad ampio spettro, cioè senza avere un'idea precisa di quali possibili errori si potranno trovare. Per evitare incomprensioni ogni modifica verrà discussa con gli autori e concordata tra le due parti.
Il *walkthrough*$_g$ è una tecnica indispensabile nelle prime fasi di sviluppo, in cui non si ha un'idea precisa dei possibili errori. Usando questa tecnica sarà poi possibile stilare una "lista di controllo", dove verranno archiviati gli errori più frequenti. Tale lista sarà salvata nell'Appendice delle "*Norme di Progetto v4.0.0*".
Inizialmente le fasi di verifica prevederanno perlopiù l'uso della tecnica *walkthrough*$_g$. Appena la lista di controllo sarà sufficientemente ampia si userà sempre di più la tecnica di *Inspection*$_g$.

3.2.1.2.2 Inspection

L'*inspection*$_g$ si basa sulla lettura mirata dei documenti/codice, cercando gli errori segnalati nella lista di controllo. Man mano che le verifiche procedono la lista verrà estesa e l'*inspection*$_g$ sarà più efficace.

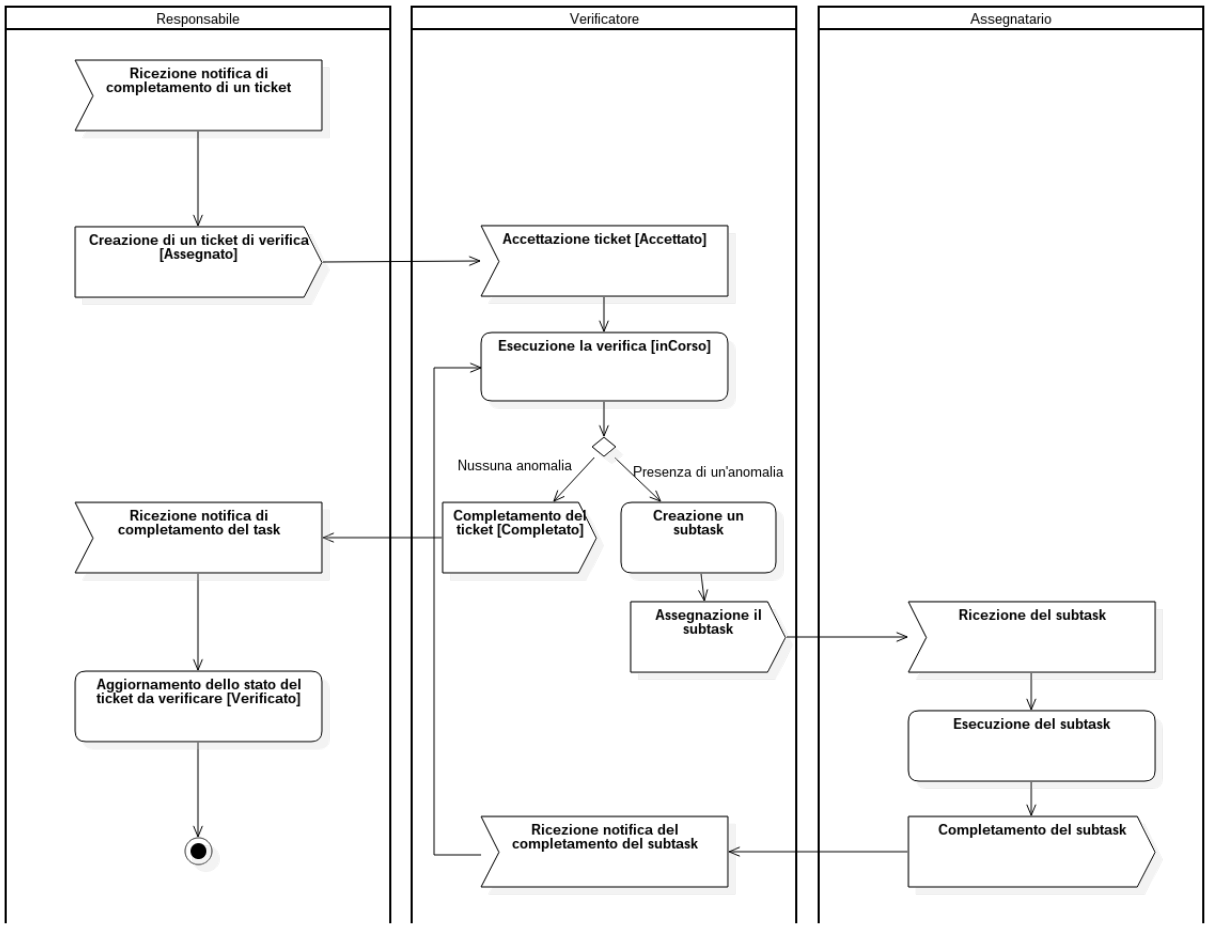

Figura 1: Ticket di verifica

3.2.1.3 Analisi dinamica

L'analisi dinamica verifica il funzionamento delle sole componenti *software*$_g$ tramite dei test e aiuta ad identificarne le anomalie.
Per garantire un risultato attendibile il test dev'essere ripetibile, ovvero deve ottenere sempre lo stesso output dallo stesso input quando viene eseguito nello stesso ambiente. Solo così il test garantisce di trovare eventuali problemi e quindi verificare la correttezza del *prodotto*$_g$ *software*$_g$.
Per ogni test deve essere definito:

- **ambiente:** è l'insieme dei sistemi *hardware*$_g$ e *software*$_g$ sui quali viene pianificata l'esecuzione del test sul *prodotto*$_g$. Va inoltre specificato uno stato iniziale da cui il test deve partire;
- **specifica:** consiste nella specifica degli input e degli output attesi;
- **procedure:** è la specifica di ulteriori istruzioni sull'esecuzione del test e sull'interpretazione dei risultati.

Nelle prossime sezioni sono definiti i tipi di test effettuati sul *prodotto*$_g$ *software*$_g$ in sviluppo dal *team*$_g$.
Nei seguenti test per **Tipo Requisito** si assume solo uno fra i seguenti valori:

- F: funzionale;
- Q: di qualità;
- P: prestazionale;
- V: di vincolo.

3.2.1.3.1 Test di unità

Il test di unità verifica che ogni singola unità di *prodotto*$_g$ funzioni correttamente. Per unità si intende una porzione di *software*$_g$ abbastanza piccola da poterne assegnare lo sviluppo ad un singolo Programmatore.
Attraverso questo test verrà verificata la correttezza di ogni modulo base che compone il *software*$_g$, limitando gli errori di implementazione.
I test di unità sono identificati dalla seguente sintassi:

TU[Codice test]

3.2.1.3.2 Test di integrazione

Il test di integrazione verifica che due o più moduli già verificati funzionino come previsto quando vengono assemblati insieme. Inoltre aiuta a scovare i difetti residui sfuggiti dalla fase di test precedente.
In più con questo test viene verificata la collaborazione tra i moduli prodotti e le componenti esterne usate come *framework*$_g$ o librerie.
È possibile che vengano usate delle componenti "fittizie" al posto dei moduli non ancora completati per poter eseguire comunque dei test.
I test di integrazione sono identificati dalla seguente sintassi:

TI[identificativo del componente]

Tale identificativo corrisponde al componente i cui elementi sono integrati.

3.2.1.3.3 Test di sistema

I test di sistema validano l'intero *prodotto*$_g$ *software*$_g$ quando si ritiene che esso sia giunto ad una versione definitiva. Viene quindi controllato se tutti i requisiti sono stati soddisfatti.
I test di sistema sono identificati dalla seguente sintassi:

TS[Tipo Requisito][Codice Requisito]

Il tipo e il codice si riferiscono al requisito di cui verrà testato il soddisfacimento.

3.2.1.3.4 Test di regressione

Questo test viene eseguito dopo la modifica di una componente, e consiste nel rieseguire tutti i test. In questo modo si controlla che dopo la modifica tutti i moduli continuino a funzionare correttamente.
Il tracciamento aiuta a capire quali sono i test da ripetere (di ogni tipo) poiché potenzialmente a rischio in caso di modifica.

3.2.1.3.5 Test di validazione

Coincide con il collaudo del *software*$_g$ in presenza del *proponente*$_g$. Se il test risulta positivo il *prodotto*$_g$ sarà considerato sufficientemente maturo da permetterne il rilascio.
I test di validazione sono identificati dalla seguente sintassi:

TV[Tipo Requisito][Codice Requisito]

Il tipo e il codice si riferiscono al requisito di cui verrà testato il soddisfacimento.

3.2.1.4 Tracciamento

Sarà compito dei Verificatori controllare la corrispondenza di ogni requisito con una o più fonti. In caso di anomalie bisogna comportarsi come descritto nella sezione 3.2.2.1.

3.2.2 Procedure

3.2.2.1 Gestione delle anomalie

Il **validatore** qualora riscontrasse delle anomalie procederà secondo la seguente procedura:

- crea un nuovo *subtask*$_g$ del *ticket*$_g$ di verifica per ogni anomalia riscontrata;
- assegna un titolo breve e preciso ad ogni *subtask*$_g$;
- se necessario aggiunge un commento che descriva l'anomalia riscontrata;
- assegna i subtasks al redattore del documento.

3.2.3 Strumenti

3.2.3.1 Controllo ortografico automatico

Grazie all'utilizzo di *TeXstudio*$_g$ è possibile, utilizzando il dizionario condiviso sul *Google Drive*$_g$ del team, correggere in fase di scrittura, gli errori ortografici che vengono automaticamente segnalati tramite la sottolineatura rossa. Per attivare questa funzione è necessario accedere alla scheda "Opzioni" nel menù orizzontale, cliccare su "Configura TeXstudio" e sucessivamente aggiungere il dizionario nella scheda "Controllo linguistico".

3.2.3.2 Indice Gulpease

È stato creato dal team uno script in grado di calcolare l'indice di *Gulpease*$_{g}$ di tutti i documenti. Tale indice sancisce la validità di un documento secondo i valori specificati nel "Piano di Qualifica v4.0.0 ".

3.2.3.3 Controllo delle parole in glossario

Per verificare che tutte le parole del glossario abbiano la segnatura corretta nei documenti il team ha creato uno script che automaticamente le marca qualora ci sia stata qualche dimenticanza in fase di stesura.

3.2.3.4 Metrics Reloaded

Plug-in, creato da *JetBrains*$_{g}$, installato in *Android Studio*$_{g}$ che permette il calcolo delle metriche sul codice *Java*$_{g}$. Grazie all'utilizzo di questo strumento verranno calcolate le seguenti metriche illustrate nel "Piano di Qualifica v4.0.0 ":

- complessità ciclomatica;
- numero di linee di codice;
- numero di parametri per ogni metodo;
- livello d'annidamento;
- livello d'instabilità;
- chiamate innestate di metodi.

3.2.3.5 jsmeter

Servizio web per il calcolo delle metriche sul codice *Javascript*$_{g}$. Grazie all'utilizzo di questo strumento verranno calcolate le metriche illustrate nel "Piano di Qualifica v4.0.0 ".

3.2.3.6 Android Studio

Android studio permette la creazione dei test di unità e i test d'integrazione, inoltre sarà utilizzato per il calcolo della copertura del codice, una volta completati i test.

3.2.3.7 Travis-CI

Travis-CI è un servizio che utilizzato per effettuare il build e per testare in automatico progetti su *GitHub*$_{g}$;inoltre permette di l'interazione con *Slack*$_{g}$ per cui ad ogni push vengono effettuati i test e viene notificato tutto il team sugli esiti.

4 Processi organizzativi

4.1 Processo di gestione

4.1.1 Attività

4.1.1.1 Comunicazioni

4.1.1.1.1 Comunicazioni interne

Per le comunicazioni interne si è deciso di utilizzare *Slack*$_g$. Sono stati creati diversi canali per suddividere il lavoro. I canali sono:

- *#general*: in questo canale va utilizzato per le comunicazioni di carattere generale e per le convocazioni delle riunioni;
- *#random*: in caso di piccoli dubbi o questioni di scarsa rilevanza, si consiglia l'utilizzo di questo canale;
- *#tools*: questo canale è stato creato per domande relative a dubbi e/o problemi sugli strumenti utilizzati per lo sviluppo;
- *#git*$_g$: su questo canale è stato collegato un *bot*$_g$ che si occupa di notificare tutti i *commit*$_g$ effettuati da ogni singolo componente del gruppo;
- *#nome_del_documento*: per ogni documento è stato creato un canale dove è possibile discutere di tutte le scelte da prendere per quel singolo documento.

In caso di necessità, sarà possibile aggiungere nuovi canali nelle fasi successive dello sviluppo.
Inoltre è possibile utilizzare il gruppo creato su *Telegram*$_g$ per comunicazioni interne non inerenti al *progetto*$_g$. Per le videoconferenze invece verrà utilizzato *Google Hangouts*$_g$.

4.1.1.1.2 Comunicazioni esterne

Per le comunicazioni esterne è stata creata la casella di posta elettronica:

beaconstrips.swe@gmail.com

Questo indirizzo deve essere l'unico usato per comunicare con le componenti esterne al team ed è controllato unicamente dal Responsabile. Il Responsabile è tenuto poi ad informare i membri del team riguardo le comunicazioni avvenute con l'esterno.

4.1.1.2 Riunioni

4.1.1.2.1 Riunioni interne

Il gruppo si ritroverà ad organizzare una riunione almeno ogni dieci giorni.
Il Responsabile si occuperà di convocare su *Slack*$_g$ le riunioni generali cui tutti i membri del team sono convocati, avvisando i componenti con almeno due giorni di preavviso.
In caso di necessità un componente del team può richiedere la convocazione di una riunione: tale richiesta deve essere inoltrata al Responsabile, che deciderà se accettarla o respingerla.
Inoltre sono possibili riunioni tra specifici membri: in questo caso sono tenuti ad informare il resto del team tramite un verbale, nel caso siano state prese decisioni rilevanti.

4.1.1.2.2 Riunioni esterne

Il Responsabile si occuperà di concordare con il *proponente*$_{\text{g}}$ o con i committenti le riunioni esterne al gruppo. È gradita la presenza di tutti i componenti del gruppo ad ogni incontro.
Prima dell'inizio di ogni riunione verrà scelto un componente del gruppo che svolgerà la funzione di segretario, il quale annoterà ogni argomento trattato e si occuperà di redigere un verbale che andrà poi inviato agli altri componenti del team.

4.1.1.3 Ticket

I *ticket*$_{\text{g}}$ avranno la seguente struttura:

- **Titolo**: deve rappresentare in modo chiaro e conciso il *task*$_{\text{g}}$;
- **Scadenza**: data entro cui l'assegnatario deve concludere il *task*$_{\text{g}}$;
- **Descrizione**: se necessario,deve chiarire con una breve descrizione il *task*$_{\text{g}}$;
- **Assegnatario**: colui che dovrà completare il *task*$_{\text{g}}$;
- **Tag di stato**: deve contenere il giusto tag che rappresenta lo stato attuale del *task*$_{\text{g}}$;
- **Subtasks**: il *task*$_{\text{g}}$ può essere suddiviso in *task*$_{\text{g}}$ piu piccoli.

4.1.1.3.1 Lista dei tag

Ogni *ticket*$_{\text{g}}$ dovrà sempre contenere un tag che rappresenti lo stato attuale di avanzamento:

- **Assegnato**: il *ticket*$_{\text{g}}$ è stato assegnato ad un membro del team;
- **Accettato**: l'assegnatario ha preso in carico il *ticket*$_{\text{g}}$;
- **Rifiutato**: l'assegnatario ha rifiutato il *ticket*$_{\text{g}}$ ed ha inserito la motivazione in un commento;
- **InCorso**: l'assegnatario sta svolgendo il *task*$_{\text{g}}$;
- **InSospeso**: l'assegnatario specifica della motivazione per cui ha messo in attesa il *ticket*$_{\text{g}}$;
- **Completato**: l'assegnatario ha concluso il *task*$_{\text{g}}$;
- **Verificato**: il *ticket*$_{\text{g}}$ è stato verificato.

4.1.2 Procedure

4.1.2.1 Creazione di un *ticket*$_{\text{g}}$

La creazione e l'assegnazione dei *ticket*$_{\text{g}}$ sarà delegata al Responsabile che seguirà la seguente procedura:

- assegna il **titolo** del *ticket*$_{\text{g}}$;
- assegna la data di **scadenza** del *ticket*$_{\text{g}}$;
- se necessario descrive il *task*$_{\text{g}}$ più approfonditamente tramite la **descrizione**;
- assegna il *ticket*$_{\text{g}}$ ad un **assegnatario**;
- imposta il **tag di stato** ad **Assegnato**.

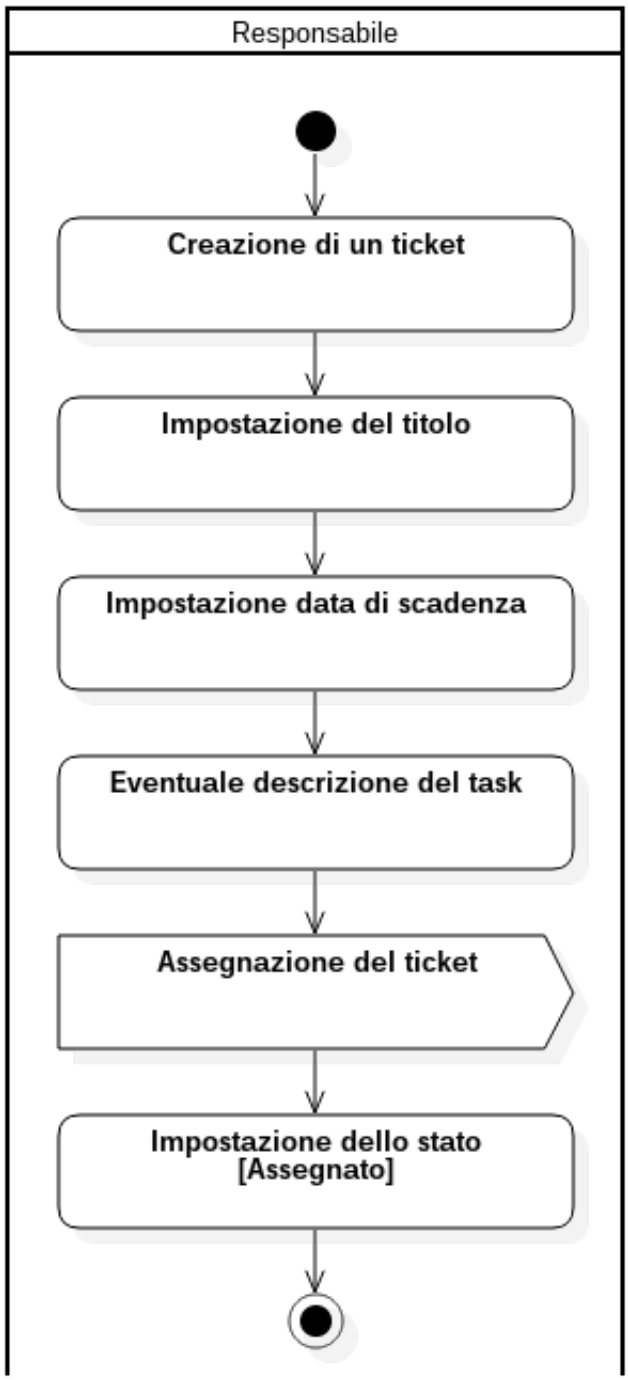

Figura 2: Creazione di un *ticket*$_g$

4.1.2.2 Aggiornamento di un *ticket*$_g$

Ogni membro del gruppo avrà una lista di *ticket*$_g$ a lui assegnati, su di essi dovrà modificare il tag di stato ogni qual volta uno di questi cambi di stato come descritto dalla sezione 4.1.1.3.1. Nel caso in cui il *ticket*$_g$ venga **rifiutato** o venga messo **in sospeso** è necessario scrivere una motivazione attraverso un commento.

4.1.3 Norme

4.1.3.1 *Repository*$_g$

Per la gestione condivisa e per il controllo delle versioni dei file il *team*$_g$ ha deciso di utilizzare una *repository*$_g$ fornita dal servizio web *GitHub*$_g$.

4.1.3.1.1 Struttura del *repository*$_g$

La respository si divide in due cartelle principali:

- **Codice**: contenente il codice del *progetto*$_g$, con struttura ancora da definire;
- **Documenti**: contenente:
  1. ***template*$_g$**: con al suo interno i file `.sty` e una cartella **img** contenente le immagini (vedi sezione 3.1.2.1);
  2. ***script*$_g$**: con al suo interno gli *script*$_g$;
  3. una cartella per ogni **revisione di avanzamento**, il cui nome è definito dal numero e dalla sigla della revisione, separata da un trattino. Ciascuna di esse è suddivisa in:
     (a) **interni**: contenente una cartella per ogni documento interno, con nome, in notazione *CamelCase*$_g$, uguale a quello del documento(senza numero di versione);
     (b) **esterni**: contenente una cartella per ogni documento esterno, con nome, in notazione *CamelCase*$_g$, uguale a quello del documento(senza numero di versione);

     Ogni cartella dei documenti ha al suo interno, se necessario, una cartella **img** contenente le immagini e i diagrammi presenti in quel documento.

4.1.3.1.2 Tipi di file e .gitignore

All'interno delle cartelle dei documenti saranno presenti solamente i file `.tex`, i restanti file ausiliari prodotti da *LaTeX*$_g$ sono stati aggiunti a .gitignore e quindi vengono ignorati e resi invisibili da *Git*$_g$.

4.1.3.1.3 Norme sulla *commit*$_g$

Per rendere effettive le modifiche applicate ai file della *repository*$_g$ locale è necessario eseguire una ***commit*$_g$**. I membri del team sono tenuti ad effettuare una *commit*$_g$ ogni qual volta venga aggiunta, rimossa o aggiornata una funzionalità oppure una sezione, aggiungendo i file interessati tramite il commando:

git$_g$ `add nome_del_file.est`

ed eseguendo la *commit*$_g$ tramite il commando:

git$_g$ *commit*$_g$ `-m ''messaggio''`

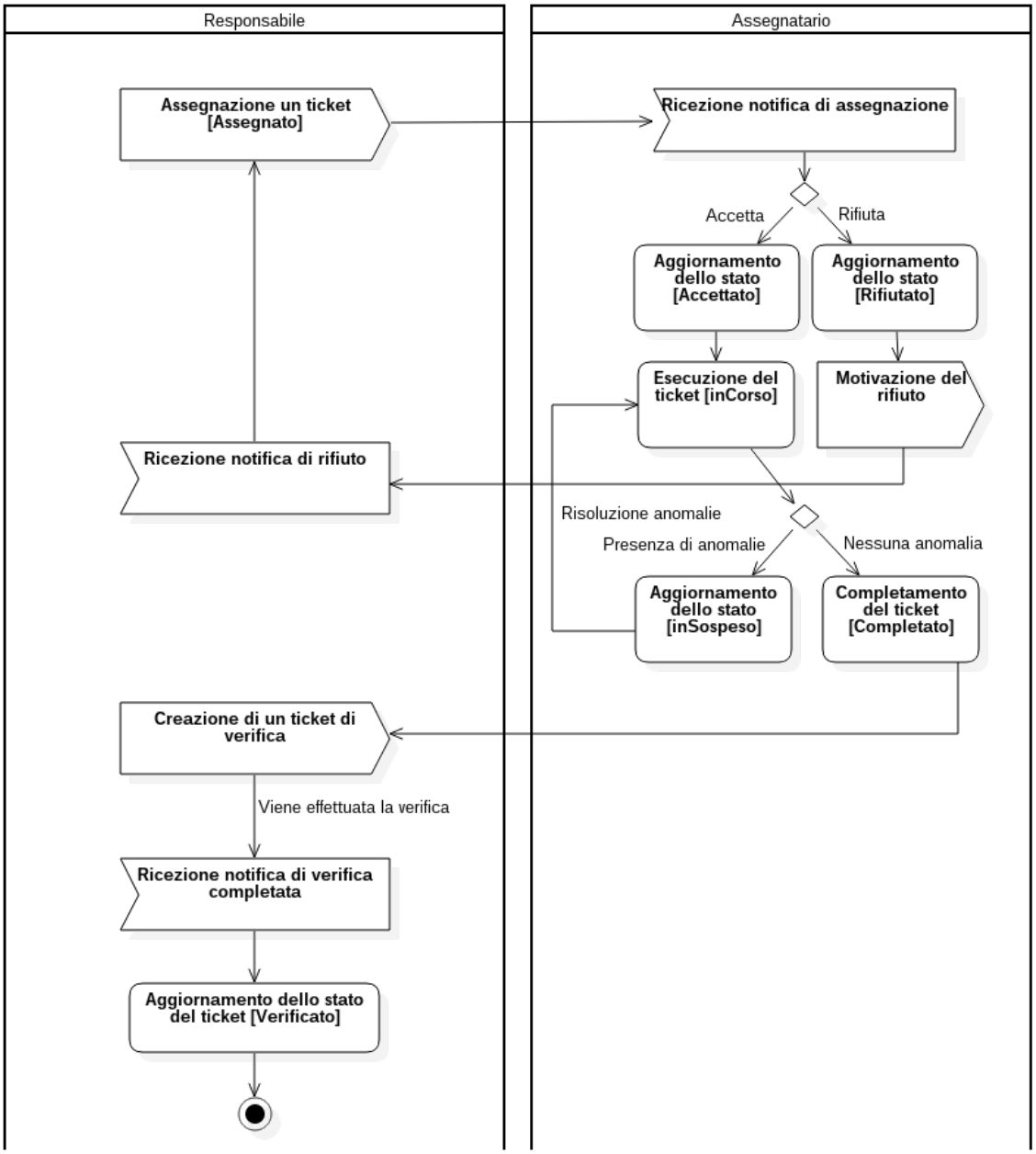

Figura 3: Ciclo di vita di un $ticket_g$

con `messaggio` che segua le regole della sezione 3.2.1. Infine per rendere effettive le modifiche nel *repository*$_g$ in remoto eseguire:

$$git_g\ push_g$$

4.1.3.1.4 Messaggi delle *commit*$_g$

Ogni qual volta venga eseguita una $commit_g$ è obbligatorio inserire un breve messaggio con la spiegazione delle modifiche apportate; il messaggio deve iniziare con:

- **Add:** se vengono aggiunte funzionalità o sezioni;
- **Remove:** se vengono tolte funzionalità o sezioni;
- **Update:** se vendono aggiornate funzionalità o sezioni;
- **Fix:** se viene sistemato un bug noto;

deve seguire una descrizione sintetica in lingua italiana delle modifiche apportate.

4.1.4 Strumenti

4.1.4.1 Sistema operativo

Non ci sono restrizioni riguardo l'utilizzo di un sistema operativo: ogni membro del team può utilizzare quindi il sistema operativo che preferisce; in ogni caso, è tenuto a rendere il materiale *prodotto*$_g$ **completamente** compatibile su tutte le piattaforme.

4.1.4.2 Condivisione

Il servizio da utilizzare per i documenti informali è *Google Drive*$_g$. All'interno di questo *repository*$_g$ vanno caricati documenti che:

- non necessitano di controllo di versione;
- necessitano di essere modificati contemporaneamente da più persone; infatti *Google Drive*$_g$ permette la modifica di un documento in contemporanea: in questo modo non ci saranno conflitti.

È possibile anche condividere link o manuali utili per la formazione del team. Si può installare *Google Drive*$_g$ sul proprio PC: i file così potranno essere disponibili anche offline e sincronizzare il contenuto della cartella ad ogni aggiornamento.

4.1.4.3 Gestione

- **Ticket**: per la gestione dei *ticket*$_g$ è stato scelto di utilizzare *Asana*$_g$; per la gestione dei *ticket*$_g$, si rimanda alla sezione 4.1.2.1;
- **Requisiti e casi d'uso**: per il tracciamento dei requisiti si utilizzerà *Trender*$_g$. Per quanto riguarda la rappresentazione dei casi d'uso l'editor consigliato è *Star UML*$_g$;
- **Scadenze**: *Google Calendar*$_g$ verrà utilizzato per segnare le scadenze del gruppo; si riceverà una notifica su *Slack*$_g$ trenta minuti prima dell'evento;
- **Progetto**: per la realizzazione dei grafici di *Gantt*$_g$ è stato impiegato il programma *Gantt-Project*$_g$.