# CLIPS

Communication & Localization with Indoor Positioning Systems

---

## Università di Padova

## Norme di progetto v6.00


leaf.gruppo@gmail.com

| | |
|---|---|
| **Versione** | 6.00 |
| **Data Redazione** | 2016-05-23 |
| **Redazione** | Oscar Elia Conti |
| **Verifica** | Davide Castello |
| **Approvazione** | Eduard Bicego |
| **Uso** | Interno |
| **Distribuzione** | *Leaf* |

# Diario delle modifiche

| Versione | Data | Autore | Ruolo | Descrizione |
|---|---|---|---|---|
| 6.00 | 2016-05-23 | Eduard Bicego | Responsabile di progetto | Approvazione del documento |
| 5.02 | 2016-05-19 | Davide Castello | Verificatore | Verifica delle ultime modifiche |
| 5.01 | 2016-05-18 | Oscar Elia Conti | Amministratore | Modifica sezione Aggiornamento del repository clips e correzioni minori |
| 5.00 | 2016-04-23 | Davide Castello | Responsabile di progetto | Approvazione del documento |
| 4.05 | 2016-04-22 | Andrea Tombolato | Verificatore | Verifica del documento |
| 4.04 | 2016-04-21 | Eduard Bicego | Amministratore | Aggiornata sezione Configurazione |
| 4.03 | 2016-04-20 | Eduard Bicego | Amministratore | Aggiunti strumenti al processo di sviluppo |
| 4.02 | 2016-04-19 | Eduard Bicego | Amministratore | Modificato paragrafo Struttura del repository |
| 4.01 | 2016-04-19 | Eduard Bicego | Amministratore | Aggiunto paragrafo Kit di sviluppo |
| 4.00 | 2016-03-20 | Oscar Elia Conti | Responsabile di progetto | Approvazione del documento |
| 3.06 | 2016-03-19 | Marco Zanella | Verificatore | Verifica del documento |
| 3.05 | 2016-03-19 | Davide Castello | Amministratore | Correzioni ortografiche |
| 3.04 | 2016-03-19 | Marco Zanella | Verificatore | Verifica del documento |

| Versione | Data | Autore | Ruolo | Descrizione |
|---|---|---|---|---|
| 3.02 | 2016-03-16 | Davide Castello | Amministratore | Aggiunta procedura per il calcolo valori delle metriche con Android Studio |
| 3.01 | 2016-03-16 | Davide Castello | Amministratore | Aggiunta procedura e strumenti per calcolo del consuntivo (sezioni 4.1.5.2 e 4.1.6.3) |
| 3.00 | 2016-02-26 | Marco Zanella | Responsabile di progetto | Approvazione del documento |
| 2.16 | 2016-02-26 | Davide Castello | Verificatore | Verifica del documento |
| 2.16 | 2016-02-25 | Federico Tavella | Amministratore | Correzioni ortografiche varie |
| 2.15 | 2016-02-25 | Davide Castello | Verificatore | Verifica del documento |
| 2.14 | 2016-02-24 | Federico Tavella | Amministratore | Stesura procedure tracciamento componenti e classi |
| 2.13 | 2016-02-24 | Andrea Tombolato | Amministratore | Aggiunti strumenti per le metriche (sezione 3.2.1.3) |
| 2.12 | 2016-02-24 | Federico Tavella | Amministratore | Aggiunta norma classificazione dei test (sezione 2.2.3.6.1) |
| 2.11 | 2016-02-23 | Federico Tavella | Amministratore | Aggiunta sezione Progettazione (sezione 2.2.2) |

| Versione | Data | Autore | Ruolo | Descrizione |
|---|---|---|---|---|
| 2.10 | 2016-02-23 | Federico Tavella | Amministratore | Aggiunta norma IDE (sezione 2.2.5.4) |
| 2.09 | 2016-02-23 | Andrea Tombolato | Amministratore | Aggiunta procedura commit (sezione 3.3.1.3.3) |
| 2.08 | 2016-02-23 | Federico Tavella | Amministratore | Stesura scheletro nuove sezioni |
| 2.07 | 2016-02-22 | Federico Tavella | Amministratore | Aggiunto strumento per la generazione del diario delle modifiche (sezione 3.1.5.2) |
| 2.06 | 2016-02-22 | Federico Tavella | Amministratore | Aggiunta norma sezione Formazione (sezione 4.2) |
| 2.05 | 2016-02-22 | Federico Tavella | Amministratore | Creazione diagrammi delle attività per i ticket (figura 9) |
| 2.04 | 2016-02-22 | Andrea Tombolato | Amministratore | Modifica norme per i ticket (sezione 4.1.4.1 e 4.1.4.2) |
| 2.03 | 2016-02-22 | Federico Tavella | Amministratore | Aggiunte norme per il codice (sezione 2.2.3) |
| 2.02 | 2016-02-22 | Federico Tavella | Amministratore | Aggiunta norma per il riferimento di una decisione (sezione 3.1.3.4.2) |

| Versione | Data | Autore | Ruolo | Descrizione |
|---|---|---|---|---|
| 2.01 | 2016-02-22 | Federico Tavella | Amministratore | Aggiunta norma per la nomenclatura del verbali (sezione 3.1.3.4.1) |
| 2.00 | 2016-02-20 | Cristian Andrighetto | Responsabile di progetto | Approvazione del documento |
| 1.12 | 2016-02-20 | Federico Tavella | Verificatore | Verficia ultime correzioni |
| 1.11 | 2016-02-20 | Davide Castello | Amministratore | Correzione errori riguardanti l'utilizzo dei comandi LaTeX nelle sezioni 3.1.3.1, 3.1.3.2 e 3.1.3.3 |
| 1.10 | 2016-02-20 | Eduard Bicego | Amministratore | Sostituzione norma per indicare i documenti sezione 3.1.2.5.8 |
| 1.09 | 2016-02-20 | Davide Castello | Amministratore | Aggiunta descrizione estesa per sigle CLIPS, BLE e IPS nelle sezioni 1.1 e 1.2 |
| 1.08 | 2016-02-19 | Davide Castello | Amministratore | Correzione errori ortografici nelle sezioni 2.1.1, 2.2.4.1, 2.2.4.1, 3.1.1, 3.1.2.1.3, 3.1.2.4.1, 4.2.2.1, 4.2.2.2, 4.2.3 |

| Versione | Data | Autore | Ruolo | Descrizione |
|---|---|---|---|---|
| 1.07 | 2016-02-19 | Eduard Bicego | Amministratore | Tolte lettere maiuscole dagli elenchi sezioni: 2, 3.1.2.5.4, 3.2.1.*, 3.3, 4.2, 4.2.1.1, 4.2.3, 4.3.*, 4.5.2.1 |
| 1.06 | 2016-02-19 | Federico Tavella | Verificatore | Verifica del documento |
| 1.05 | 2016-02-18 | Eduard Bicego | Amministratore | Strutturazione dei processi in attività |
| 1.04 | 2016-02-18 | Eduard Bicego | Amministratore | Stesura processo di validazione |
| 1.03 | 2016-02-18 | Davide Castello | Amministratore | Aggiunti diagrammi delle attività per ogni procedura |
| 1.02 | 2016-02-17 | Davide Castello | Amministratore | Aggiunto processo di fornitura nei processi primari e spostata la sezione studio di fattibilità |
| 1.01 | 2016-02-17 | Davide Castello | Amministratore | Rimossa la parola processo da tutti i titoli di secondo livello |
| 1.00 | 2016-01-12 | Federico Tavella | Responsabile di progetto | Approvazione del documento |
| 0.15 | 2016-01-11 | Eduard Bicego | Verificatore | Verifica del documento |
| 0.15 | 2016-01-10 | Oscar Elia Conti | Amministratore | Divisione processi in norme, procedure e strumenti |

| Versione | Data | Autore | Ruolo | Descrizione |
|---|---|---|---|---|
| 0.14 | 2016-01-09 | Marco Zanella | Amministratore | Ristrutturazione del documento per processi |
| 0.13 | 2015-12-08 | Oscar Elia Conti | Amministratore | Correzione errori rilevati in fase di verifica |
| 0.12 | 2015-12-07 | Eduard Bicego | Verificatore | Verifica del documento |
| 0.11 | 2015-12-06 | Oscar Elia Conti | Amministratore | Aggiunta strumenti UML, tracciamento requisiti e di pianificazione |
| 0.10 | 2015-12-05 | Marco Zanella | Amministratore | Stesura introduzione documento |
| 0.09 | 2015-12-04 | Oscar Elia Conti | Amministratore | Stesura norme riguardanti le riunioni |
| 0.08 | 2015-12-03 | Marco Zanella | Amministratore | Stesura norme riguardanti studio di fattibilità e analisi dei requisiti |
| 0.07 | 2015-12-02 | Marco Zanella | Amministratore | Stesura norme riguardanti i ruoli di progetto |
| 0.06 | 2015-12-01 | Marco Zanella | Amministratore | Stesura norme riguardanti il controllo di versione |
| 0.05 | 2015-11-30 | Marco Zanella | Amministratore | Stesura norme riguardanti studio di fattibilità e analisi dei requisiti |

| Versione | Data | Autore | Ruolo | Descrizione |
|---|---|---|---|---|
| 0.04 | 2015-11-30 | Oscar Elia Conti | Amministratore | Terminata stesura norme riguardanti la documentazione |
| 0.03 | 2015-11-29 | Oscar Elia Conti | Amministratore | Inizio stesura norme riguardanti la documentazione |
| 0.02 | 2015-11-28 | Oscar Elia Conti | Amministratore | Stesura norme riguardanti la struttura dei documenti |
| 0.01 | 2015-11-27 | Marco Zanella | Amministratore | Stesura struttura documento e norme riguardanti la verifica |

# Indice

**1 Introduzione** **1**
1.1 Scopo del documento . . . 1
1.2 Scopo del prodotto . . . 1
1.3 Glossario . . . 1
1.4 Riferimenti utili . . . 2
1.4.1 Riferimenti normativi . . . 2
1.4.2 Riferimenti informativi . . . 2

**2 Processi primari** **3**
2.1 Fornitura . . . 3
2.1.1 Studio di fattibilità . . . 3
2.2 Sviluppo . . . 3
2.2.1 Analisi dei requisiti . . . 3
2.2.1.1 Classificazione dei casi d'uso . . . 3
2.2.1.2 Classificazione dei requisiti . . . 4
2.2.2 Progettazione . . . 5
2.2.2.1 Progettazione architetturale . . . 5
2.2.2.1.1 Design pattern . . . 6
2.2.2.1.2 Diagrammi UML . . . 6
2.2.2.1.3 Classificazione dei componenti . . . 7
2.2.2.1.4 Test di integrazione . . . 7
2.2.2.2 Progettazione di dettaglio . . . 7
2.2.2.2.1 Design pattern . . . 8
2.2.2.2.2 Diagrammi UML . . . 8
2.2.2.2.3 Classificazione di una classe . . . 9
2.2.2.2.4 Descrivere una classe . . . 9
2.2.2.2.5 Test di unità . . . 9
2.2.3 Codifica . . . 10
2.2.3.1 Convenzioni . . . 10
2.2.3.2 Nomi . . . 10
2.2.3.3 Ricorsione . . . 10
2.2.3.4 Documentazione . . . 10
2.2.3.4.1 File . . . 10
2.2.3.4.2 Classi . . . 10
2.2.3.4.3 Metodi . . . 11
2.2.3.4.4 Casi eccezionali . . . 11
2.2.3.5 Versionamento dei file di codice . . . 11
2.2.3.6 Test . . . 11
2.2.3.6.1 Classificazione dei test . . . 12

2.2.4 Procedure . . . 12
2.2.4.1 Tracciamento componenti-requisiti . . . 12
2.2.4.1.1 Inserimento di un componente su Tracy . . . 12
2.2.4.2 Tracciamento classi-requisiti . . . 13
2.2.4.2.1 Inserimento di una classe su Tracy . 13
2.2.5 Strumenti . . . 13
2.2.5.1 Creazione dei diagrammi UML . . . 13
2.2.5.2 Tracciamento dei requisiti . . . 13
2.2.5.3 Tracciamento dei test . . . 14
2.2.5.4 Scrittura del codice . . . 14
2.2.5.5 Kit di sviluppo . . . 14
2.2.5.6 Esecuzione dei test . . . 14
2.2.5.7 Continuous integration . . . 14
2.2.5.8 Analisi statica del codice . . . 14

**3 Processi di supporto 15**
3.1 Documentazione . . . 15
3.1.1 Struttura dei documenti . . . 15
3.1.1.1 Frontespizio . . . 15
3.1.1.2 Informazioni sul documento . . . 16
3.1.1.3 Diario delle modifiche . . . 16
3.1.1.4 Indice delle sezioni . . . 17
3.1.1.5 Indice delle tabelle . . . 17
3.1.1.6 Indice delle figure . . . 17
3.1.1.7 Introduzione . . . 17
3.1.1.8 Contenuto . . . 17
3.1.2 Norme tipografiche . . . 18
3.1.2.1 Formattazione generale . . . 18
3.1.2.1.1 Testatine . . . 18
3.1.2.1.2 Piè pagina . . . 18
3.1.2.1.3 Orfani e vedove . . . 18
3.1.2.2 Caratteri . . . 18
3.1.2.2.1 Virgolette . . . 18
3.1.2.2.2 Parentesi . . . 19
3.1.2.2.3 Punteggiatura . . . 19
3.1.2.2.4 Numeri . . . 19
3.1.2.3 Stile del testo . . . 19
3.1.2.3.1 Corsivo . . . 19
3.1.2.3.2 Grassetto . . . 19
3.1.2.3.3 Sottolineato . . . 20

3.1.2.3.4 Monospace . . . . . 20
3.1.2.3.5 Glossario . . . . . 20
3.1.2.4 Composizione del testo . . . . . 20
3.1.2.4.1 Elenchi . . . . . 20
3.1.2.4.2 Descrizioni . . . . . 21
3.1.2.4.3 Note a piè pagina . . . . . 21
3.1.2.5 Formati . . . . . 21
3.1.2.5.1 Date . . . . . 21
3.1.2.5.2 Orari . . . . . 21
3.1.2.5.3 URI . . . . . 22
3.1.2.5.4 Sigle . . . . . 22
3.1.2.5.5 Ruoli di progetto . . . . . 23
3.1.2.5.6 Fasi del progetto . . . . . 23
3.1.2.5.7 Revisioni . . . . . 23
3.1.2.5.8 Nomi . . . . . 23
3.1.2.6 Componenti grafiche . . . . . 24
3.1.2.6.1 Immagini . . . . . 24
3.1.2.6.2 Tabelle . . . . . 24
3.1.3 Tipologie di documenti . . . . . 24
3.1.3.1 Documenti formali . . . . . 24
3.1.3.2 Documenti informali . . . . . 25
3.1.3.3 Glossario . . . . . 25
3.1.3.4 Verbali . . . . . 25
3.1.3.4.1 Nome del verbale . . . . . 26
3.1.3.4.2 Verbali di riunioni interne . . . . . 26
3.1.3.4.3 Verbali di riunioni esterne . . . . . 27
3.1.4 Procedure . . . . . 27
3.1.4.1 Versionamento dei documenti . . . . . 27
3.1.4.2 Avanzamento di un documento . . . . . 27
3.1.4.2.1 Regole di avanzamento di versione . 28
3.1.4.2.2 Formalizzazione di un documento . . 28
3.1.5 Strumenti . . . . . 28
3.1.5.1 Latex . . . . . 28
3.1.5.1.1 Template . . . . . 28
3.1.5.1.2 Comandi personalizzati . . . . . 30
3.1.5.1.3 Rilevamento errori ortografici . . . . 30
3.1.5.2 Diario delle modifiche . . . . . 30
3.2 Qualità . . . . . 30
3.2.1 Metriche . . . . . 30
3.2.1.1 Classificazione delle metriche . . . . . 30
3.2.1.2 Procedure . . . . . 31

3.2.1.2.1 Calcolo del valore di una metrica con Android Studio . . . 31
3.2.1.3 Strumenti . . . 32
3.2.1.3.1 Copertura requisiti obbligatori . . . 32
3.2.1.3.2 Copertura requisiti desiderabili . . . 32
3.2.1.3.3 Numero di statement per metodo . . . 32
3.2.1.3.4 Numero di campi dati per classe . . . 32
3.2.1.3.5 Grado di accoppiamento . . . 32
3.2.1.3.6 Cyclomatic number . . . 32
3.2.1.3.7 Numero di parametri per metodo . . . 32
3.2.1.3.8 Adequacy of variable name . . . 32
3.2.1.3.9 Average module size . . . 32
3.2.1.3.10 Test passati richiesti . . . 32
3.2.1.3.11 Failure avoidance . . . 33
3.2.1.3.12 Breakdown avoidance . . . 33
3.2.2 Obiettivi . . . 33
3.2.2.1 Classificazione degli obiettivi . . . 33
3.3 Configurazione . . . 34
3.3.1 Controllo di versione . . . 34
3.3.1.1 Richieste di modifica . . . 34
3.3.1.2 Repository . . . 34
3.3.1.2.1 Struttura del repository Documents . 34
3.3.1.2.2 Struttura del repository clips . . . 35
3.3.1.2.3 Commit . . . 35
3.3.1.2.4 Visibilità del repository . . . 36
3.3.1.2.5 File . . . 36
3.3.1.2.5.1 Nomi dei file . . . 36
3.3.1.2.5.2 Codifica dei file . . . 36
3.3.1.3 Procedure . . . 37
3.3.1.3.1 Procedura di richiesta di modifica . . 37
3.3.1.3.2 Aggiornamento del repository Documents . . . 38
3.3.1.3.3 Aggiornamento del repository clips . 41
3.3.1.3.4 Esecuzione di un commit . . . 41
3.3.1.4 Strumenti . . . 42
3.4 Verifica . . . 43
3.4.1 Documenti . . . 43
3.4.1.1 Sintassi . . . 43
3.4.1.2 Periodi . . . 43
3.4.1.3 Struttura del documento . . . 43
3.4.1.4 Procedure . . . 43

3.4.1.4.1 Resoconto attività di verifica sui documenti . . . 44
3.4.1.4.1.1 Attività manuale di verifica . 44
3.4.1.4.1.2 Attività automatica di verifica 44
3.4.2 Diagrammi UML . . . 45
3.4.2.1 Diagrammi dei casi d'uso . . . 45
3.4.2.2 Procedure . . . 45
3.4.2.2.1 Procedura di verifica attori dei diagrammi UML . . . 45
3.4.3 Issue tracking . . . 46
3.4.3.1 Sintassi di una label . . . 46
3.4.3.2 Sintassi di una Issue . . . 47
3.4.3.3 Procedure . . . 48
3.4.3.3.1 Gestione di una issue . . . 48
3.4.3.4 Strumenti . . . 53
3.4.3.4.1 Strumento per l'issue tracking . . . 53
3.4.4 Analisi . . . 53
3.4.4.1 Analisi statica . . . 53
3.4.4.2 Analisi dinamica . . . 53
3.5 Validazione . . . 54
3.5.1 Responsabilità . . . 54
3.5.2 Procedure . . . 54
3.5.2.1 Procedura per la validazione . . . 54

**4 Processi organizzativi 55**
4.1 Gestione organizzativa . . . 55
4.1.1 Comunicazione . . . 55
4.1.1.1 Comunicazioni interne . . . 55
4.1.1.2 Comunicazioni esterne . . . 55
4.1.1.3 Composizione delle email . . . 55
4.1.1.3.1 Mittente . . . 56
4.1.1.3.2 Destinatario . . . 56
4.1.1.3.3 Oggetto . . . 56
4.1.1.3.4 Corpo . . . 56
4.1.1.3.5 Allegati . . . 56
4.1.2 Riunioni . . . 57
4.1.2.1 Riunioni interne . . . 57
4.1.2.1.1 Convocazione di una riunione interna 57
4.1.2.1.2 Gestione di una riunione interna . . 58
4.1.2.2 Riunioni esterne . . . 58
4.1.2.2.1 Convocazione di una riunione esterna 58

4.1.2.2.2 Gestione di una riunione esterna . . 58
4.1.2.3 Verbale di una riunione . . . . . 59
4.1.3 Ruoli di progetto . . . . . 59
4.1.3.1 Responsabile di progetto . . . . . 60
4.1.3.2 Amministratore . . . . . 60
4.1.3.3 Analista . . . . . 61
4.1.3.4 Progettista . . . . . 61
4.1.3.5 Programmatore . . . . . 62
4.1.3.6 Verificatore . . . . . 62
4.1.3.7 Rotazione dei ruoli . . . . . 62
4.1.4 Ticketing . . . . . 63
4.1.4.1 Task list . . . . . 63
4.1.4.2 Ticket . . . . . 63
4.1.5 Procedure . . . . . 63
4.1.5.1 Procedura di creazione e gestione di un ticket 63
4.1.5.2 Stesura del consuntivo . . . . . 66
4.1.6 Strumenti . . . . . 67
4.1.6.1 Pianificazione . . . . . 67
4.1.6.2 Creazione dei diagrammi di Gantt . . . . . 68
4.1.6.3 Calcolo del consuntivo . . . . . 68
4.2 Formazione . . . . . 69
4.2.1 Formazione dei membri del gruppo . . . . . 69
4.2.1.1 Ore rendicontabili e di investimento . . . . . 69

# Elenco delle figure

1 Procedura di formalizzazione di un documento . . . . . 29
2 Procedura di richiesta modifica di un documento . . . . . 38
3 Procedura di aggiornamento del repository Documents . . . . 40
4 Procedura di commit sul repository . . . . . 42
5 Procedura di verifica di un attore . . . . . 46
6 Procedura di gestione di una issue già aperta . . . . . 49
7 Procedura di gestione di una issue. Parte 1 . . . . . 51
8 Procedura di gestione di una issue. Parte 2 . . . . . 52
9 Procedura di creazione e gestione di un ticket - Parte 1 . . . . 64
10 Procedura di creazione e gestione di un ticket - Parte 2 . . . . 66
11 Procedura di determinazione delle ore . . . . . 69

# 1 Introduzione

## 1.1 Scopo del documento

Questo documento ha lo scopo di stabilire le norme che tutti i membri del gruppo *Leaf* dovranno rispettare durante lo sviluppo del progetto Communication & Localization with Indoor Positioning Systems(*CLIPS*$_g$), gli strumenti che dovranno utilizzare e le procedure che dovranno seguire.
Tutti i membri del gruppo sono tenuti a prendere visione di tale documento e rispettare le norme in esso contenute, al fine di garantire uniformità nello svolgimento del progetto, garantendo maggiore efficienza ed efficacia alle attività.
Il documento rivolge l'attenzione sui seguenti contenuti:

- organizzazione della comunicazione all'interno del gruppo e verso l'esterno;
- modalità di stesura dei documenti;
- descrizione delle metodologie di lavoro durante lo sviluppo del progetto;
- modalità di gestione e utilizzo del repository$_g$;
- strumenti per la gestione dell'ambiente di lavoro.

## 1.2 Scopo del prodotto

Lo scopo del prodotto$_g$ è implementare un metodo di navigazione indoor$_g$ che sia funzionale alla tecnologia Bluetooth$_g$ Low Energy(BLE$_g$). Il prodotto$_g$ comprenderà un prototipo software$_g$ che permetta la navigazione all'interno di un'area predefinita, basandosi sui concetti di Indoor Positioning System(IPS$_g$) e smart places$_g$.

## 1.3 Glossario

Allo scopo di rendere più semplice e chiara la comprensione dei documenti viene allegato il *Glossario v5.00* nel quale verranno raccolte le spiegazioni di terminologia tecnica o ambigua, abbreviazioni ed acronimi. Per evidenziare un termine presente in tale documento, esso verrà marcato con il pedice $_g$.

## 1.4 Riferimenti utili

### 1.4.1 Riferimenti normativi

- Capitolati$_g$ d'appalto:
  http://www.math.unipd.it/~tullio/IS-1/2015/Progetto/Capitolati.html

### 1.4.2 Riferimenti informativi

- Materiale di riferimento del corso di Ingegneria del software$_g$:
  - http://www.math.unipd.it/~tullio/IS-1/2015/Dispense/L01.pdf;
  - http://www.math.unipd.it/~tullio/IS-1/2015/Dispense/L04.pdf;
  - http://www.math.unipd.it/~tullio/IS-1/2015/Dispense/L05.pdf;
  - http://www.math.unipd.it/~tullio/IS-1/2015/Dispense/L11.pdf;
- Portable Document Format:
  http://en.wikipedia.org/wiki/Portable_Document_Format;
- Rappresentazione dei numeri:
  https://en.wikipedia.org/wiki/ISO_31-0#Numbers;
- Rappresentazione di date e orari:
  https://en.wikipedia.org/wiki/ISO_8601;
- Unicode$_g$:
  http://en.wikipedia.org/wiki/Unicode;
- Bluetooth$_g$ low energy:
  https://en.wikipedia.org/wiki/Bluetooth_low_energy;
- Tracy$_g$:
  https://github.com/dontpanic-swe/tracy.

# 2 Processi primari

## 2.1 Fornitura

### 2.1.1 Studio di fattibilità

Il documento riguardante lo studio di fattibilità deve essere redatto rapidamente ed in modo accurato dagli *Analisti* sulla base di ciò che è emerso nelle prime riunioni, nelle quali si deve discutere di temi riguardanti i capitolati, come:

- rischi nell'affrontare ogni capitolato$_g$;
- rapporto tra i costi ed i benefici, sia in base al mercato attuale che futuro, sia in base al costo di produzione e alla possibile redditività futura;
- il dominio applicativo e tecnologico di ogni capitolato$_g$.

## 2.2 Sviluppo

### 2.2.1 Analisi dei requisiti

L'*Analisi dei requisiti* è il documento dove devono essere catalogati e descritti tutti i requisiti che il prodotto$_g$ finale deve soddisfare. Ogni requisito deve emergere da una delle seguenti fonti:

- capitolati$_g$ d'appalto;
- incontri con il proponente;
- incontri con il committente;
- valutazioni effettuate durante riunioni interne al gruppo.

Tale documento deve inoltre riportare il modo in cui ogni requisito deve essere verificato.

**2.2.1.1 Classificazione dei casi d'uso** È compito degli *Analisti* redigere una descrizione, dare una classificazione e fornire un diagramma conforme allo standard UML$_g$ per ogni caso d'uso. Ogni caso d'uso dev'essere descritto con le seguenti informazioni, possibilmente in quest'ordine:

1. codice identificativo del caso d'uso, nella forma

**UC[X].[Y]**

dove:

- **X** è il codice univoco del padre;
- **Y** è un codice progressivo di livello.

Il codice progressivo può includere diversi livelli di gerarchia separati da un punto.

2. titolo, che deve descrivere sinteticamente il caso d'uso;
3. attori principali;
4. attori secondari, se questi sono presenti;
5. precondizioni, ovvero le condizioni che necessariamente devono verificarsi prima del caso d'uso;
6. postcondizioni, ciò che deve essere verificato successivamente al caso d'uso;
7. flusso principale degli eventi, dove si descrive il flusso dei casi d'uso figli. Per ogni evento va specificato:
   - una descrizione testuale dell'evento;
   - gli attori coinvolti;
   - se l'azione è descritta dettagliatamente da un altro caso d'uso.
8. Scenari alternativi, ovvero scenari in cui si verificano eccezioni o errori. Per ognuno di questi deve essere indicato:
   - una descrizione testuale dell'evento;
   - gli attori coinvolti;
   - se l'azione è descritta dettagliatamente da un altro caso d'uso.

**2.2.1.2 Classificazione dei requisiti** È compito degli *Analisti* redigere e classificare i requisiti. I requisiti devono essere classificati in base al tipo e alla priorità, utilizzando la seguente notazione:

**R[X][Y][Z]**

, dove

1. **X** indica l'importanza strategica del requisito. Deve assumere solo i seguenti valori:
   - **Obb**: Indica un requisito obbligatorio;
   - **Des**: Indica un requisito desiderabile;
   - **Opz**: Indica un requisito opzionale.
2. **Y** indica la tipologia del requisito. Deve assumere solo i seguenti valori:
   - **F**: Indica un requisito funzionale;
   - **Q**: Indica un requisito di qualità;
   - **P**: Indica un requisito prestazionale;
   - **V**: Indica un requisito vincolo.
3. **Z** rappresenta il codice univoco di ogni requisito in forma gerarchica.

### 2.2.2 Progettazione

Lo scopo dell'attività di progettazione è generare una soluzione soddisfacente per tutti gli individui che fanno parte del progetto. 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$_g$ finito.

**2.2.2.1 Progettazione architetturale** Per poter arrivare alla definizione dell'architettura dei vari prodotti software$_g$, è necessario passare attraverso una serie di attività, quali:

1. individuare i prodotti che si intendono realizzare;
2. definire ruoli e responsabilità per ogni prodotto$_g$ individuato;
3. definire le interazioni che i prodotti hanno fra di loro;
4. assicurarsi che ogni requisito sia soddisfatto da almeno uno dei prodotti individuati;
5. ci si assicura che ogni prodotto$_g$ soddisfi almeno un requisito.

A questo punto, è possibile procedere con la progettazione dell'architettura dei prodotti software$_g$ che devono essere realizzati. Si devono eseguire e documentare i seguenti task$_g$ per ogni prodotto$_g$ individuato:

1. suddividere il prodotto$_g$ in componenti;
2. definire il ruolo di ogni componente individuato;
3. definire le interazioni tra i vari componenti;
4. definire le interfacce che ogni componente mette a disposizione dell'ambiente esterno;
5. assicurarsi che ogni requisito sia soddisfatto da almeno uno dei componenti
6. assicurarsi che ogni componente soddisfi almeno un requisito;
7. realizzare e documentare i test di integrazione, al fine di verificare il corretto funzionamento di più componenti integrati assieme.

Durante la suddivisione del prodotto$_g$ in più componenti i *Progettisti* dovranno utilizzare i Design pattern$_g$ e, nel farlo, dovranno provvedere a descriverli.

**2.2.2.1.1 Design pattern** Per migliorare la comprensibilità delle scelte progettuali e della progettazione stessa i *Progettisti* dovranno indicare i pattern architetturali utilizzati, fornendo per ciascuno d'essi:

- descrizione testuale;
- descrizione grafica;
- motivazione dell'utilizzo;
- descrizione di come viene applicato il pattern al progetto.

**2.2.2.1.2 Diagrammi UML** Per documentare prodotti e componenti individuati, sarà necessario ricorrere a dei diagrammi UML$_g$, necessari per formalizzare gli aspetti descritti in modo testuale. Durante la progettazione architetturale, sarà necessario ricorrere a

**diagrammi dei componenti:** hanno lo scopo di rappresentare la struttura interna di un prodotto$_g$ software$_g$ modellato in termini dei suoi componenti principali e delle relazioni fra di essi. Questo tipo di diagramma viene dunque utilizzato per evidenziare i componenti (e le relazioni che essi hanno fra di loro) individuati all'interno di ciascun prodotto$_g$ software$_g$;

**diagrammi delle attività:** hanno lo scopo di definire le attività da svolgere per realizzare una certa funzionalità. Vengono dunque usati per mostrare come determinate interazioni tra componenti (o fra prodotti) realizzino una funzionalità che si intende rendere disponibile.

**2.2.2.1.3 Classificazione dei componenti** I componenti vengono identificati univocamente da una descrizione nella forma

**NomeProdotto::NomeComponente**

dove:

- **NomeProdotto** è il nome del prodotto$_g$ software$_g$ che i *Progettisti* hanno individuato;
- **NomeComponente** rappresenta il nome assegnato al componente dai *Progettisti*.

Inoltre, ogni componente ha associato un codice univoco nella forma

**[X][Y]**

dove:

- **X** è l'iniziale del nome del prodotto$_g$;
- **Y** è un numero intero incrementale.

**2.2.2.1.4 Test di integrazione** Per ottimizzare l'attività di test, i *Progettisti* devono preoccuparsi di definire delle classi di verifica per accertare il funzionamento dei componenti. L'ideale sarebbe fornire degli strumenti automatici ai *Verificatori*. La progettazione delle classi per la verifica deve essere svolta nel rispetto della mancata sovrapposizione dei ruoli: chi crea una classe non dovrebbe essere lo stesso individuo che crea la classe per testarla.

**2.2.2.2 Progettazione di dettaglio** Una volta definita l'architettura del sistema, si procede alla progettazione di dettaglio, la quale prevede che i seguenti task$_g$:

1. individuare le classi che implementano ciascuno dei componenti individuati mediante la progettazione architetturale;
2. definire i ruoli e le responsabilità di ogni classe individuata;

3. definire nel dettaglio ogni classe;
4. assicurarsi che ogni requisito sia soddisfatto da almeno di una delle classi definite;
5. assicurarsi che ogni classe soddisfi almeno un requisito;
6. realizzare e documentare i test di unità, necessari per verificare il corretto comportamento di ogni classe.

**2.2.2.2.1 Design pattern** Per migliorare la comprensibilità delle scelte progettuali e della progettazione stessa i *Progettisti* dovranno indicare i pattern architetturali utilizzati, fornendo per ciascuno d'essi:

- descrizione testuale;
- descrizione grafica;
- motivazione dell'utilizzo;
- descrizione di come viene applicato il pattern al progetto.

**2.2.2.2.2 Diagrammi UML** Per documentare prodotti e componenti individuati sarà necessario ricorrere a dei diagrammi $\text{UML}_g$, necessari per formalizzare gli aspetti descritti in modo testuale. Durante la progettazione architetturale, sarà necessario ricorrere a

**diagrammi delle classi:** hanno lo scopo di descrivere i tipi di entità, con le loro caratteristiche ed eventuali relazioni fra questi tipi. Questo diagramma viene utilizzato per evidenziare e descrivere in modo dettagliato le classi e le relazioni che esistono fra di loro;

**diagrammi delle attività:** hanno lo scopo di definire le attività da svolgere per realizzare una certa funzionalità. Vengono dunque usati per mostrare come determinate interazioni tra componenti (o fra prodotti) realizzino una funzionalità che si intende rendere disponibile;

**diagramma di sequenza** hanno lo scopo di descrivere scenari e vengono usati per descrivere le relazioni che intercorrono, in termini di messaggi, tra attori, oggetti ed entità del sistema rappresentato.

**2.2.2.2.3 Classificazione di una classe** Le classi vengono univocamente identificate da una descrizione nella forma

**NomeProdotto::NomeComponente::NomeClasse**

dove

- **NomeProdotto** è il nome del prodotto$_g$ software$_g$ che i *Progettisti* hanno individuato;
- **NomeComponente** rappresenta il nome assegnato al componente dai *Progettisti*;
- **NomeClasse** rappresenta il nome che è stato dato dai *Progettisti* alla classe individuata.

Inoltre, ad ogni classe è associato un codice univoco nella forma

**[X][Y]**

dove

- **X** è l'iniziale del nome del prodotto$_g$;
- **Y** è un numero intero incrementale.

**2.2.2.2.4 Descrivere una classe** Per descrivere una classe, i *Progettisti* devono preoccuparsi di descrivere:

- nome;
- visibilità;
- attributi;
- metodi;
- descrizione generica che indichi lo scopo e la responsabilità della classe.

Inoltre, devono essere descritte dettagliatamente (anche attraverso diagrammi) le relazioni con altre classi e le interfacce messe a disposizione.

**2.2.2.2.5 Test di unità** Per la realizzazione dei test di unità, i *Programmatori* devono rispettare la regola di mancata sovrapposizione dei ruoli: chi realizza il test non deve essere lo stesso individuo che realizza la classe da testare.

### 2.2.3 Codifica

**2.2.3.1 Convenzioni** Tutti i file contenenti codice o testo dovranno rispettare la codifica UTF-8 senza BOM$_g$.
I *Programmatori*, dovendo sviluppare un'applicazione Android$_g$, dovranno seguire le indicazioni fornite dalla guida Google$_g$ Java$_g$ Style[1].

**2.2.3.2 Nomi** I nomi di variabili, metodi e classi dovranno essere in notazione CamelCase$_g$ e in lingua inglese. I nomi di variabili e metodi dovranno avere la prima lettera minuscola, mentre per le classi sarà maiuscola.

**2.2.3.3 Ricorsione** La ricorsione dovrà essere evitata il più possibile, limitando il suo uso ai soli casi indispensabili, ossia dove sia dimostrato che non è possibile utilizzare un metodo iterativo con complessità computazionale minore. Per ogni metodo ricorsivo, dovrà essere fornita una prova della sua terminazione.

#### 2.2.3.4 Documentazione

**2.2.3.4.1 File** I file contenenti codice dovranno avere un'intestazione contenente:

```
/**
 * @author Nome dell'autore
 * @version Versione corrente del file
 * @since Versione del file nel momento
 *    dell'aggiunta al progetto
 *
 * Descrizione del file
 */
```

**2.2.3.4.2 Classi** Poiché in Java$_g$ è presente una forte dipendenza tra file e classi (ogni classe in Java$_g$ corrisponde ad un file .class), non è necessario aggiungere un ulteriore intestazione. Per le classi interne e quelle anonime dovrà essere aggiunta un'ulteriore intestazione come segue:

```
/**
 * Descrizione della classe
 */
```

[1] https://google.github.io/styleguide/javaguide.html

**2.2.3.4.3 Metodi** Per ogni metodo, dovrà essere presente un'intestazione contenente:

```
/**
 * Descrizione del metodo
 * @param NomeParametro1 Descrizione del primo
    parametro
 * ...
 * @param NomeParametroN Descrizione del N-esimo
    parametro
 * @return TipoDiRitorno Valore ritornato dal metodo
 * @throws TipoDiEccezione Motivo di lancio
    dell'eccezione
 */
```

Per i costruttori la voce `@return` deve essere omessa.

**2.2.3.4.4 Casi eccezionali** Qualora fosse necessario documentare parti di codice di difficile comprensione è permesso l'utilizzo di un commento nelle righe precedenti, che dovrà essere strutturato come segue:

```
/**
 * Descrizione del blocco di codice
 */
```

**2.2.3.5 Versionamento dei file di codice** Il numero di versione dei file di codice appare solamente nell'intestazione di ciascuno di essi. Deve seguire il formato:

**[X].[Y]**

, dove

- **X** è un numero intero, incrementale, corrispondente all'ultima versione stabile del codice;
- **Y** è un numero intero, incrementale, corrispondente al numero di modifiche apportate al codice, partendo dall'ultima versione stabile.

**2.2.3.6 Test**

**2.2.3.6.1 Classificazione dei test** È compito dei *Programmatori* creare test di unità e integrazione e degli *Analisti* creare quelli sistema e accettazione al fine di verificare che tutti i requisiti individuati dagli *Analisti* siano soddisfatti. Ogni test deve essere codificato come:

**T[X][Y]**

, dove

1. **X** rappresenta il tipo di test. Può assumere solo i seguenti valori:
   - **A**: indica un test di accettazione;
   - **S**: indica un test di sistema;
   - **I**: indica un test di integrazione;
   - **U**: indica un test di unità.
2. **Y** rappresenta il codice univoco di ogni test in forma gerarchica.

### 2.2.4 Procedure

**2.2.4.1 Tracciamento componenti-requisiti** Per tracciare un requisito su un componente dell'applicazione è sufficiente applicare la seguente procedura su Tracy$_g$:

1. aprire l'homepage di Tracy$_g$;
2. selezionare la voce "Requirements";
3. individuare il requisito che si desidera tracciare e cliccare su "View";
4. sotto la voce "Packages" indicare il nome del package$_g$ e cliccare su "Add".

**2.2.4.1.1 Inserimento di un componente su Tracy** Per inserire un componente all'interno di Tracy$_g$:

1. aprire l'homepage di Tracy$_g$;
2. selezionare la voce "Package";
3. dal menu "Operations" selezionare la voce "Create package";
4. riempire almeno i campi obbligatori della form e cliccare su "Create".

**2.2.4.2 Tracciamento classi-requisiti** Per tracciare un requisito su una classe dell'applicazione è sufficiente applicare la seguente procedura su Tracy$_g$:

1. aprire l'homepage di Tracy$_g$;
2. selezionare la voce "Requirements";
3. individuare il requisito che si desidera tracciare e cliccare su "View";
4. sotto la voce "Classes" indicare il nome della classe e cliccare su "Add".

**2.2.4.2.1 Inserimento di una classe su Tracy** Per inserire una classe all'interno di Tracy$_g$:

1. aprire l'homepage di Tracy$_g$;
2. selezionare la voce "Class";
3. dal menu "Operations" selezionare la voce "Create class";
4. riempire almeno i campi obbligatori della form e cliccare su "Create".

### 2.2.5 Strumenti

**2.2.5.1 Creazione dei diagrammi UML** Lo strumento per la creazione dei diagrammi UML$_g$ utilizzato è Astah$_g$.

**2.2.5.2 Tracciamento dei requisiti** Lo strumento scelto per la il tracciamento dei requisiti è Tracy$_g$. Questo software$_g$ è stato sviluppato dal gruppo di Ingegneria del software$_g$ Don't Panic. Il software$_g$ è stato scelto per le seguenti caratteristiche:

- open source$_g$;
- tracciamento dei requisiti;
- tracciamento use case;
- tracciamento delle fonti;
- stesura automatica in LaTeX$_g$ dei requisiti.

Poiché questo software$_g$ non risulta essere completamente perfetto, il gruppo ha previsto di riadattarlo sulla base delle esigenze che sono emerse durante la stesura dell'*Analisi dei requisiti*.

**2.2.5.3 Tracciamento dei test** Per il tracciamento dei test, verrà utilizzato Tracy$_g$. Il software$_g$ è stato scelto in modo da non dover utilizzare diverse piattaforme per il tracciamento dei requisiti e dei test.

**2.2.5.4 Scrittura del codice** Per lo sviluppo dell'applicazione Android$_g$, il gruppo ha deciso di utilizzare l'IDE$_g$ Android Studio$_g$ per i seguenti motivi:

- gratis;
- multipiattaforma;
- IDE$_g$ di riferimento per lo sviluppo di applicazioni Android$_g$;
- possibilità di simulare un dispositivo;
- integrabile con GitHub$_g$;
- basato su IntelliJ Idea$_g$, strumento già utilizzato dai membri del team$_g$.

La versione utilizzata dal gruppo è la 1.5.1.

**2.2.5.5 Kit di sviluppo** Per la codifica si utilizzano gli strumenti offerti dalla JDK (Java Development Kit$_g$) versione 1.8. In particolare:

- javac: il compilatore per il linguaggio Java.
- Java Virtual Machine: macchina virtuale creata per eseguire il codice java compilato.

**2.2.5.6 Esecuzione dei test** Per la creazione e l'esecuzione dei test di unità e di integrazione, è possibile utilizzare, in combinazione con Android Studio$_g$, il Framework$_g$ JUnit$_g$.

**2.2.5.7 Continuous integration** Per l'integrazione continua del codice caricato nel repository$_g$ dai membri del gruppo si utilizza lo strumento Travis CI$_g$ che permette l'esecuzione di tutti i test ad ogni *pull request* richiesta da un *Programmatore*. Inoltre rende disponibile una dashboard$_g$ da cui poter fruire della cronologia di ogni build effettuata ad ogni *pull request* richiesta con il relativo esito.

**2.2.5.8 Analisi statica del codice** Per effettuare l'analisi statica del codice scritto è possibile utilizzare FindBugs$_g$ direttamente in Android Studio$_g$ tramite il plug-in FindBugs-IDEA. In alternativa è possibile utilizzare Lint$_g$ strumento di analisi statica implementato in Android Studio$_g$.

# 3 Processi di supporto

## 3.1 Documentazione

In questa sezione sono indicati gli standard riguardanti la struttura e la stesura dei documenti prodotti.

### 3.1.1 Struttura dei documenti

Ogni documento è realizzato a partire da una struttura prestabilita che dovrà essere uguale per tutti i documenti ufficiali ad eccezione dei verbali:

1. frontespizio;
2. informazioni sul documento;
3. diario delle modifiche;
4. indice delle sezioni;
5. indice delle tabelle;
6. indice delle figure;
7. introduzione;
8. contenuto.

L'ordine di ognuna delle sezioni è fissato. La numerazione delle prime pagine sarà quella romana, mentre dall'introduzione fino alla fine del documento quella araba.

**3.1.1.1 Frontespizio** Questa sezione deve trovarsi nella prima pagina di ogni documento e contiene:

1. informazioni sul gruppo:
   a. nome;
   b. logo;
   c. email.
2. informazioni sul progetto:
   a. nome progetto;
   b. nome azienda proponente.

3. informazioni sul documento:
    a. nome;
    b. versione.

**3.1.1.2 Informazioni sul documento** In questa sezione vengono indicate le principale informazioni riguardanti il documento quali:

1. versione;
2. data di redazione;
3. cognome e nome di coloro che hanno redatto il documento (in ordine alfabetico);
4. cognome e nome di coloro che hanno verificato il documento (in ordine alfabetico);
5. ambito d'uso del documento (interno oppure esterno);
6. cognome e nome di coloro ai quali è destinato il documento (in ordine alfabetico).

**3.1.1.3 Diario delle modifiche** Questa sezione descrive, attraverso l'utilizzo di una tabella, le modifiche che sono state apportate al documento. Ogni riga della tabella corrisponde ad una modifica effettua al documento. La struttura della riga della tabella è la seguente:

1. versione del documento;
2. data della modifica;
3. cognome e nome dell'autore della modifica;
4. ruolo dell'autore della modifica nel momento in cui essa è avvenuta;
5. descrizione delle modifiche apportate.

Le righe della tabella sono ordinate a partire dalla data dell'ultima modifica effettuata, in ordine cronologico inverso.

**3.1.1.4 Indice delle sezioni** L'indice delle sezioni contiene l'indice di tutti gli argomenti trattati all'interno del documento. La sua struttura è la seguente:

1. titolo dell'argomento trattato;
2. numero di pagina.

**3.1.1.5 Indice delle tabelle** Questa sezione contiene l'indice delle tabelle. Per ogni tabella deve essere specificato:

1. titolo della tabella;
2. numero di pagina di riferimento.

Nel caso in cui non siano presenti tabelle all'interno del documento, è possibile omettere questa sezione.

**3.1.1.6 Indice delle figure** In questa sezione sono riportate tutte le figure presenti all'interno del documento. Per ogni figura deve essere specificato:

- nome figura;
- pagina di riferimento della figura.

Nel caso in cui non siano presenti figure all'interno del documento, è possibile omettere questa sezione.

**3.1.1.7 Introduzione** Questa sezione deve riportare le seguenti informazioni:

1. scopo del documento;
2. glossario;
3. riferimenti utili:
   a. riferimenti normativi;
   b. riferimenti informativi.

**3.1.1.8 Contenuto** Questa sezione descrive il contenuto del documento. Anch'esso deve essere propriamente diviso in sezioni e sottosezioni.

### 3.1.2 Norme tipografiche

#### 3.1.2.1 Formattazione generale

**3.1.2.1.1 Testatine** Ogni pagina di un documento, fatta eccezione per il frontespizio, deve contenere la testina. Essa è composta da:

- logo del gruppo, posizionato in alto a sinistra;
- nome del documento, posizionato in alto a destra.

**3.1.2.1.2 Piè pagina** Ogni pagina di un documento, deve contenere il piè pagina. Esso contiene:

- numero della pagina, posizionato al centro.

**3.1.2.1.3 Orfani e vedove** Si considera vedova, la riga di un paragrafo che inizia alla fine di una pagina, mentre si considera orfana, la riga di un paragrafo che finisce all'inizio di una pagina. I documenti dovranno essere redatti cercando di evitare il più possibile queste due tipologie di righe poiché risultano poco gradevoli.

#### 3.1.2.2 Caratteri

##### 3.1.2.2.1 Virgolette

- **Virgolette alte singole ‘ ’ :** devono essere utilizzate per racchiudere un singolo carattere;
- **Virgolette alte doppie “ ” :** devono essere utilizzate per racchiudere:
  - nomi di file;
  - comandi;
  - collegamenti a sezioni interne dello stesso documento;
  - parole a cui è stato dato un significato particolare;
  - parole a cui è stato dato un senso diverso da quello originale.
- **Virgolette basse ‘«’ ‘»’ :** devono essere utilizzate per racchiudere citazioni.

Non sono ammessi ulteriori casi d'uso per le virgolette.

#### 3.1.2.2.2 Parentesi

- **Tonde:** possono essere utilizzate per descrivere esempi, per fornire dei sinonimi oppure per dare delle precisazioni. Sono, insieme alle parentesi quadre, le uniche parentesi ammesse all'interno di una frase.
- **Quadre:** possono rappresentare uno standard ISO oppure uno stato relativo ad un ticket$_g$.

**3.1.2.2.3 Punteggiatura** La punteggiatura deve essere sempre utilizzata attentamente per cercare di rendere il discorso il più chiaro e coeso possibile. Non sono ammesse spaziature prima dell'utilizzo di un carattere di punteggiatura. L'utilizzo del punto è necessario per indicare la fine di un concetto e poter iniziarne un altro.

**3.1.2.2.4 Numeri** I numeri all'interno dei documenti devono essere formattati seguendo lo standard [SI/ISO 31-0]. Esso prevede che la parte frazionaria sia separata da quella decimale utilizzando la virgola. I numeri la cui parte intera supera le tre cifre, devono essere scritti raggruppando in gruppi di tre le cifre di cui è composta la parte intera, partendo dalla cifra meno significativa e separandoli con uno spazio unificatore.

### 3.1.2.3 Stile del testo

**3.1.2.3.1 Corsivo** Il corsivo va utilizzato per riportare le seguenti informazioni:

- nome di un documento;
- nome di un ruolo;
- percorsi di cartelle.

**3.1.2.3.2 Grassetto** Il grassetto va utilizzato per riportare le seguenti informazioni:

- titoli;
- parole su cui è utile focalizzare l'attenzione del lettore all'interno di un argomento;
- parole chiave all'interno di elenchi.

**3.1.2.3.3 Sottolineato** La sottolineatura è indicata qualora si voglia evidenziare l'importanza di una parola all'interno di una frase.

**3.1.2.3.4 Monospace** Lo stile Monospace$_g$ va applicato nel caso in cui si vogliano riportare all'interno di un documento comandi oppure parti di codice.

**3.1.2.3.5 Glossario** Questo stile va applicato per tutte le parole che hanno una corrispondenza all'interno del *Glossario*. Ogni parola presente nel *Glossario* deve essere seguita da un pedice contente il carattere 'g' scritto in corsivo. Non si applica questa regola nei casi in cui la parola compaia all'interno di titoli, percorsi, nomi di cartelle, comandi o parti di codice.

#### 3.1.2.4 Composizione del testo

**3.1.2.4.1 Elenchi** Le norme che regolano un elenco sono le seguenti:

- la prima parola di un elenco deve essere maiuscola, fatta eccezione nel caso in cui l'elenco inizi con il carattere ':';
- ogni elemento dell'elenco, tranne l'ultimo, deve terminare con il carattere ';'. È fatta eccezione nel caso in cui l'elemento sia composto da molte frasi, allora è permesso il '.';
- l'ultimo elemento di un elenco deve sempre terminare con il carattere '.'.

È necessario usare elenchi numerati quando l'ordine degli elementi è rilevante. Per gli elenchi numerati valgono le seguenti regole:

- nel primo livello si usano numeri interni a partire da uno;
- nel secondo livello si usano lettere dell'alfabeto a partire dalla 'a'.

Gli elenchi puntati servono per descrivere elementi di cui non è importante l'ordine espositivo. Essi seguono le seguenti regole:

- nel primo livello bisogna utilizzare cerchi neri pieni;
- nel secondo livello trattini neri.

**3.1.2.4.2 Descrizioni** Nel caso in cui si voglia strutturare la descrizione di qualcosa nella forma di elenco è necessario utilizzare il costrutto LaTeX `\begin{`*description*`}` `\item[elemento]` `\end{`*description*`}`. All'interno delle parentesi quadre viene inserito il nome dell'elemento che si vuole descrivere mentre, dopo di esse, la sua descrizione.

**3.1.2.4.3 Note a piè pagina** Le note a piè pagina seguono le seguenti regole:

- la loro numerazione è progressiva all'interno di tutto il documento;
- devono essere scritte una volta sola;
- il primo carattere di ogni nota deve essere maiuscolo. Fanno eccezione i casi in cui la parola sia un acronimo. In questo caso bisogna seguire le regole di formattazione di tale acronimo.

#### 3.1.2.5 Formati

**3.1.2.5.1 Date** La formattazione delle date segue lo standard [ISO 8601]. Tale standard prevede che una data sia scritta secondo il seguente formalismo: AAAA-MM-GG. Questa rappresentazione va letta nel seguente modo:

- AAAA: numero a quattro cifre che rappresenta l'anno;
- MM: numero a due cifre che rappresenta il mese;
- GG: numero a due cifre che rappresenta il giorno.

Nei casi in cui risulti possibile esprimere i mesi o giorni omettendo una cifra, è necessario anteporre uno zero davanti a tale cifra. Per velocizzare la scrittura delle date è stato creato il comando LaTeX `\frmdate{giorno}{mese}{anno}`.

**3.1.2.5.2 Orari** La formattazione degli orari segue lo standard [ISO 8601]. Tale standard prevede che gli orari siano scritti secondo il seguente formalismo: hh:mm. Questa rappresentazione va letta nel seguente modo:

- hh: numero a due cifre che rappresenta il numero di ore trascorse dalla mezzanotte;
- mm: numero a due cifre che rappresenta i minuti.

Nei casi in cui risulti possibile esprimere le ore o i minuti omettendo una cifra, è necessario anteporre uno zero davanti a tale cifra. Per velocizzare la scrittura degli orari è stato creato il comando LaTeX `\frmtime{ora}{minuti}`.

**3.1.2.5.3 URI** La stile utilizzato per rappresentare un URI è il corsivo ed il testo deve essere di colore blu. Per velocizzare la scrittura degli URI è stato creato il comando LaTeX `\frmURI{URI}`.

**3.1.2.5.4 Sigle** È possibile fare riferimento a ruoli, documenti e revisioni pianificate utilizzando le seguenti sigle:

- Rp (*Responsabile di progetto*);
- Am (*Amministratore*);
- An (*Analista*);
- Pt (*Progettista*);
- Pm (*Programmatore*);
- Ve (*Verificatore*);
- AR (*Analisi dei requisiti*);
- GL (*Glossario*);
- NP (*Norme di progetto*);
- PP (*Piano di progetto*);
- PQ (*Piano di qualifica*);
- SF (*Studio di fattibilità*);
- ST (*Specifica tecnica*);
- RR (**Revisione dei requisiti**);
- RA (**Revisione di accettazione**);
- RP (**Revisione di progettazione**);
- RQ (**Revisione di qualifica**).

L'utilizzo di tale sigle è permesso solo all'interno di:

- tabelle;
- diagrammi (immagini);
- didascalie di tabelle e immagini;
- note a piè di pagina.

**3.1.2.5.5 Ruoli di progetto** Quando si fa riferimento ad un ruolo di progetto bisogna adottare lo stile corsivo e la prima lettera deve essere maiuscola. Per velocizzare la scrittura di un ruolo è stato creato il comando LaTeX `\frmrole{Ruolo}`.

**3.1.2.5.6 Fasi del progetto** Quando si fa riferimento ad una fase$_g$ del progetto bisogna adottare lo stile grassetto e la prima lettera deve essere maiuscola. Per velocizzare la scrittura di una fase$_g$ è stato creato il comando LaTeX `\frmphase{Fase}`.

**3.1.2.5.7 Revisioni** Quando si fa riferimento ad una revisione bisogna adottare lo stile grassetto e la prima lettera deve essere maiuscola. Per velocizzare la scrittura di una revisione è stato creato il comando LaTeX `\frmrev{Revisione}`.

**3.1.2.5.8 Nomi**

- **Nome del gruppo:** quando si fa riferimento al nome del gruppo bisogna adottare lo stile corsivo e la prima lettera deve essere maiuscola. Il nome del gruppo deve essere sempre indicato tramite il comando LaTeX `\leaf`, in modo tale da avere sempre la stessa formattazione.
- **Nome del progetto:** il nome del progetto deve essere sempre indicato tramite il comando LaTeX `\progetto`, in modo tale da avere sempre la stessa formattazione.
- **Nome proprio:** ogniqualvolta si intende utilizzare un nome proprio si deve scrivere prima il nome e successivamente il cognome. Quando si fa riferimento a disporre i nomi in ordine alfabetico, i nomi devono essere scritti mettendo il cognome davanti al nome e l'ordine è dettato dalle lettere del cognome, a meno di ulteriori specificazioni.
- **Nome di un file:** i nomi dei file vanno formattati utilizzando lo stile Monospace$_g$ e devono essere racchiusi dalle doppie virgolette alte. Per velocizzare la scrittura del nome di un file è stato creato il comando LaTeX `\frmFile{File}`.
- **Nome di un documento:** quando si fa riferimento ad un documento bisogna adottare lo stile corsivo e la prima lettera deve essere maiuscola. I nomi dei documenti devono essere scritti in corsivo. Per velocizzare la scrittura di un documento è stato creato il comando LaTeX `\frmdoc{documento}`.

#### 3.1.2.6 Componenti grafiche

**3.1.2.6.1 Immagini** L'utilizzo delle immagini all'interno di un documento è regolamentato secondo quanto segue:

- i formati ammessi per le immagini sono il PNG$_g$ e il PDF$_g$;
- devono essere numerate in ordine crescente;
- devono essere seguite da una breve descrizione;
- deve essere presente un riferimento all'immagine all'interno dell'indice immagini.

**3.1.2.6.2 Tabelle** L'utilizzo delle tabelle all'interno di un documento è regolamentato secondo quanto segue:

- devono essere numerate in ordine crescente;
- devono essere seguite da una breve descrizione;
- devono essere presente un riferimento alla tabella all'interno dell'indice delle tabelle;
- si consigliano le linee verticali all'interno delle tabelle solo se queste ne dovessero aumentare la leggibilità.

Al fine di uniformare lo stile delle tabelle in tutti i documenti, è consigliabile utilizzare il template LaTeX che è stato creato, il quale contiene lo stile predefinito delle tabelle.

### 3.1.3 Tipologie di documenti

**3.1.3.1 Documenti formali** I documenti formali possono essere descritti secondo quanto segue:

- sono documenti approvati dal *Responsabile di progetto*;
- eventuali modifiche ad un documento formale, lo rendono informale;
- sono gli unici documenti che possono essere distribuiti all'esterno del team$_g$ di progetto.

**3.1.3.2 Documenti informali** I documenti informali possono essere descritti secondo quanto segue:

- sono documenti non ancora approvati dal *Responsabile di progetto*;
- possono essere distribuiti solamente all'interno del team$_g$ di progetto;
- possono essere sottoposti a revisione.

**3.1.3.3 Glossario** Il *Glossario* nasce dall'esigenza di chiarire il significato ambiguo che possono avere certe parole all'interno di determinati contesti. Al suo interno sono quindi presenti alcune parole, prese dai documenti, che hanno le seguenti caratteristiche:

- trattano argomenti tecnici;
- trattano argomenti poco conosciuti o che possono scatenare ambiguità;
- rappresentano delle sigle.

Il *Glossario* deve essere strutturato secondo quanto segue:

- i termini devono seguire l'ordine lessicografico;
- ogni termine deve essere seguito da una spiegazione chiara e concisa del significato del termine stesso. Questa spiegazione non deve essere in alcun modo ambigua.

Per evitare confusione, la stesura del *Glossario* deve avvenire in maniera parallela alla stesura dei documenti . Al fine di evitare dimenticanze, è ammesso inserire un termine all'interno del *Glossario* senza inserirne immediatamente la spiegazione. È comunque doveroso completare la spiegazione non appena possibile.
All'interno dei documenti i termini presenti nel *Glossario* devono essere marcati con il pedice contente il carattere 'g' scritto in corsivo. (Regole per la formattazione di termini nel Glossario).

**3.1.3.4 Verbali** Lo scopo di un verbale è di riassumere, cercando di essere il più possibile fedeli, ciò che che è stato discusso durante un incontro. È previsto che ad ogni riunione tra i membri del gruppo e/o soggetti esterni sia redatto un verbale. Il verbale è soggetto ad un'unica stesura e non può subire modifiche.

**3.1.3.4.1 Nome del verbale** Il nome del verbale deve essere nella forma

**Riunione[X][Y]**

, dove:

- **X** indica il tipo di riunione a cui si riferisce il verbale. Può assumere solo i seguenti valori:
  - **Interna**: indica il verbale di una riunione interna;
  - **Esterna**: indica il verbale di una riunione esterna.
- **Y** indica la data in cui si è svolta la riunione a cui si riferisce il verbale.

**3.1.3.4.2 Verbali di riunioni interne** Si definisce interna una riunione che coinvolge solamente i membri del gruppo. Il verbale per questo tipo di incontro è da considerarsi di carattere informale.

Il verbale deve essere redatto seguendo la seguente struttura:

1. frontespizio;
2. una sezione "Estremi del Verbale" contente le seguenti informazioni:
   a. data;
   b. luogo, a meno di incontri telematici;
   c. partecipanti.
3. introduzione contenente le motivazioni per le quali è stato richiesto l'incontro;
4. ordine del giorno;
5. verbale della riunione;
6. decisioni prese, definite come una lista numerata.

Di conseguenza, se ci si vuole riferire ad una determinata decisione presa in una riunione, è sufficiente indicare il verbale e il numero della decisione (es. RiunioneInterna2016-02-01, decisione 3).

**3.1.3.4.3 Verbali di riunioni esterne** Si definisce esterna una riunione che avviene tra i membri del team$_g$ e soggetti esterni. Il verbale redatto per questo tipo di incontro è da considerarsi come parte integrante della documentazione ufficiale e per questo motivo può avere un valore normativo o fornire nuovi requisiti. È previsto che ad ogni incontro venga nominata una persona che si occupi della sua stesura.

Il verbale deve essere redatto seguendo la seguente struttura:

1. frontespizio;
2. indice;
3. una sezione "Estremi del Verbale" contente le seguenti informazioni:
   a. data;
   b. ora;
   c. luogo, a meno di incontri telematici;
   d. partecipanti.
4. introduzione contenente le motivazioni per le quali è stato richiesto l'incontro;
5. una sezione dedicata alle domande poste e alle risposte ottenute durante l'incontro.

### 3.1.4 Procedure

**3.1.4.1 Versionamento dei documenti** Tutti i documenti, ad eccezione dei verbali, sono sottoposti a versionamento. Il versionamento prevede che la versione di un documento venga incrementata ogniqualvolta avvengano delle modifiche all'interno del documento stesso. La sintassi che indica la versione di un documento è la seguente: vX.YY. La sua interpretazione è la seguente:

- 'v' è un carattere che si riferisce alla parola versione;
- X è un numero che indica quante volte è stato formalizzato il documento;
- YY è un numero a due cifre che indica quante modifiche sono state effettuate al documento dalla sua ultima formalizzazione.

**3.1.4.2 Avanzamento di un documento**

**3.1.4.2.1 Regole di avanzamento di versione** L'avanzamento di versione avviene secondo le seguenti regole:

- X inizia da 0 e viene incrementato di una unità nel momento in cui il *Responsabile di progetto* formalizza il documento;
- YY inizia da 00 e viene incrementato di una unità ad ogni modifica che viene effettua al documento. Ogni volta che il documento viene formalizzato riparte da 00.

La prima versione di ogni documento è indicata dalla versione v0.01.

**3.1.4.2.2 Formalizzazione di un documento** 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. il *Responsabile di progetto* assegna uno o più *Verificatori* al documento i quali dovranno occuparsi di controllare la correttezza del documento stesso;
3. se i *Verificatori* riscontrano anomalie si ritorna al punto 1 altrimenti il documento viene consegnato al *Responsabile di progetto*;
4. il *Responsabile di progetto* decide se approvare il documento, e quindi formalizzarlo, oppure se rifiutarlo, e quindi ritornare al punto 1.

### 3.1.5 Strumenti

**3.1.5.1 Latex** La stesura dei documenti deve essere effettuata utilizzando il linguaggio di markup$_g$ LaTeX$_g$. È stato scelto questo strumento poiché permette una facile separazione tra formattazione e presentazione. La scelta dell'editor da utilizzare è lasciata libera ai membri del gruppo.

**3.1.5.1.1 Template** Per poter creare omogeneità tra i documenti, è stato creato un template LaTeX$_g$ nel quale sono state definite tutte le regole tipografiche da applicare al documento. Questo permette di poter scrivere i documenti senza dover tener conto della loro formattazione.

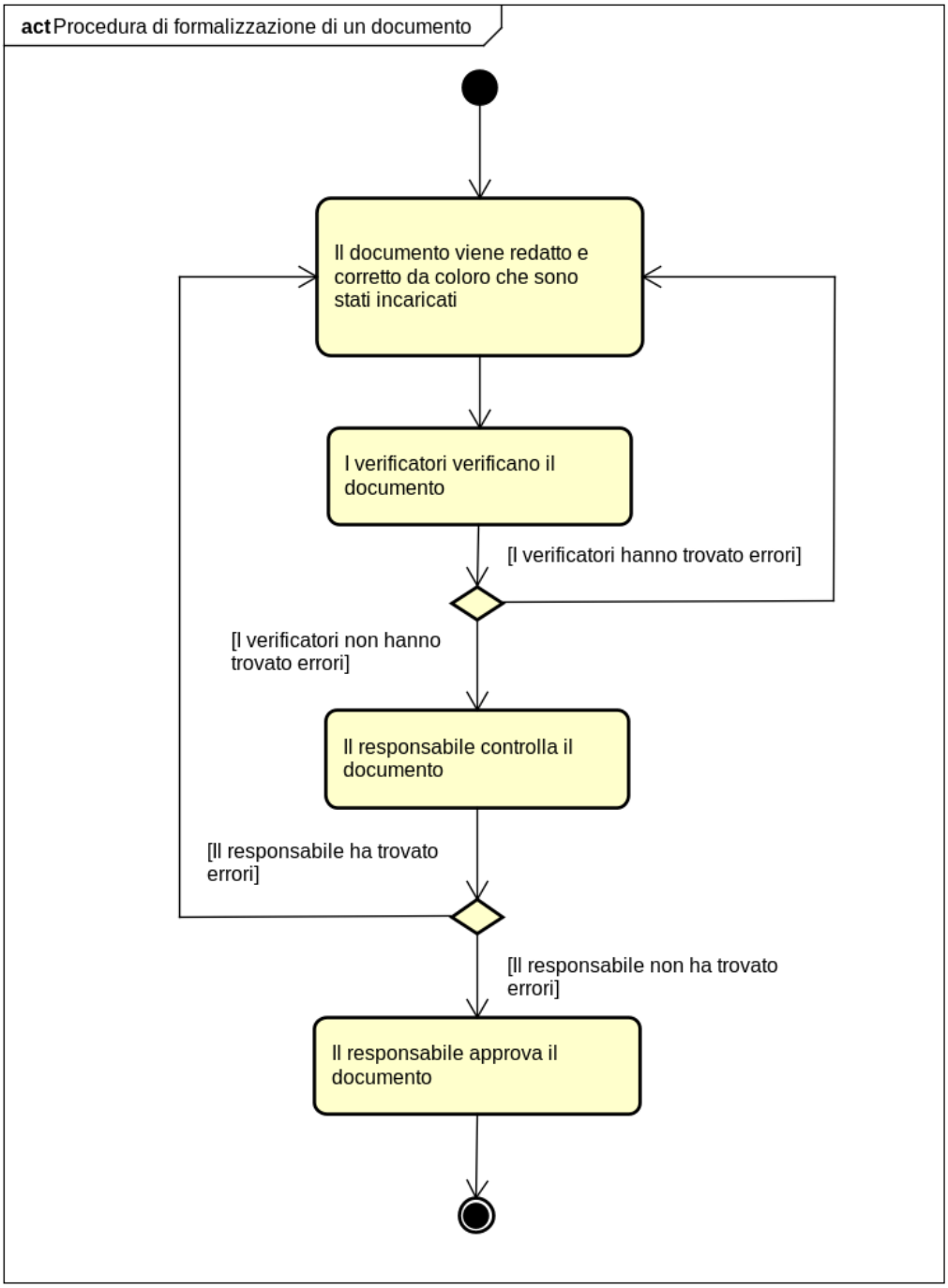


**Figura 1:** Procedura di formalizzazione di un documento

**3.1.5.1.2 Comandi personalizzati** Sono stati definiti dei comandi LaTeX$_g$ 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 nella sezione formati.

**3.1.5.1.3 Rilevamento errori ortografici** Al fine di semplificare l'operazione di individuazione e correzione degli errori ortografici, è stato creato uno script bash ‘‘`OrtographicCheck.sh`’’.

**3.1.5.2 Diario delle modifiche** Per semplificare la stesura del diario delle modifiche di ogni documento, è possibile utilizzare lo script Perl ‘‘`diaryGenerator.pl`’’ che, combinato all'hook per i commit, genera automaticamente il diario delle modifiche del documento.

## 3.2 Qualità

### 3.2.1 Metriche

**3.2.1.1 Classificazione delle metriche** Per garantire la qualità del lavoro del team$_g$ gli *Amministratori* hanno definito delle metriche, riportandole nel *Piano di qualifica*, 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;
  - **PR** 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$_g$ e può assumere i seguenti valori:
  - **D** per indicare i documenti;
  - **S** per indicare il software$_g$.
- **Z** indica il codice univoco della metrica (numero intero incrementale a partire da 1).

Riassumendo, le possibiltà per le denominazione di una metrica sono le seguenti:

- MPC[Z];
- MPRD[Z];
- MPRS[Z].

#### 3.2.1.2 Procedure

**3.2.1.2.1 Calcolo del valore di una metrica con Android Studio** Alcune metriche per il software$_g$ scelte dal gruppo necessitano, per il calcolo dei propri valori, di Android Studio$_g$ e di un suo plugin, chiamato MetricsReloaded$_g$.
Di seguito viene mostrata la procedura da seguire in queste situazioni:

1. aprire Android Studio$_g$;
2. selezionare in *Metrics scope* il box custom scope;
3. nel menu a tendina a fianco selezionare i file su cui effettuare il calcolo delle metriche;
4. selezionare in *Metrics profile* il pulsante a fianco del menu identificato da '...';
5. nella finestra ora aperta selezionare tramite checkbox la metrica da calcolare:
   - *Number of statements*: per visualizzare il numero di statement per metodo;
   - *Class size (attributes)*: per visualizzare il numero di attributi per classe;
   - *Coupling factor*: per visualizzare il grado di accoppiamento;
   - *Total cyclomatic complexity*: per visualizzare la complessità ciclomatica del codice;
   - *Number of parameters*: per visualizzare il numero di parametri per metodo;
   - *Average size*: per visualizzare la dimensione media di un modulo$_g$ in termini di linee di codice.
6. selezionare Apply e in seguito OK;
7. selezionare nuovamente OK.

#### 3.2.1.3 Strumenti

**3.2.1.3.1 Copertura requisiti obbligatori** Lo strumento scelto per il calcolo del valore di questa metrica è Tracy$_g$, il quale permette di tracciare i requisiti e mapparli su use case, moduli e classi.

**3.2.1.3.2 Copertura requisiti desiderabili** Lo strumento scelto per il calcolo del valore di questa metrica è Tracy$_g$, il quale permette di tracciare i requisiti e mapparli su use case, moduli e classi.

**3.2.1.3.3 Numero di statement per metodo** Lo strumento scelto per il calcolo del valore di questa metrica è MetricsReloaded$_g$, plugin di Android Studio$_g$.

**3.2.1.3.4 Numero di campi dati per classe** Lo strumento scelto per il calcolo del valore di questa metrica è MetricsReloaded$_g$, plugin di Android Studio$_g$.

**3.2.1.3.5 Grado di accoppiamento** Lo strumento scelto per il calcolo del valore di questa metrica è MetricsReloaded$_g$, plugin di Android Studio$_g$.

**3.2.1.3.6 Cyclomatic number** Lo strumento scelto per il calcolo del valore di questa metrica è MetricsReloaded$_g$, plugin di Android Studio$_g$.

**3.2.1.3.7 Numero di parametri per metodo** Lo strumento scelto per il calcolo del valore di questa metrica è MetricsReloaded$_g$, plugin di Android Studio$_g$.

**3.2.1.3.8 Adequacy of variable name** Lo strumento scelto per il calcolo del valore di questa metrica è Tracy$_g$, il quale permette di tracciare i requisiti e mapparli su use case, moduli e classi.

**3.2.1.3.9 Average module size** Lo strumento scelto per il calcolo del valore di questa metrica è MetricsReloaded$_g$, plugin di Android Studio$_g$.

**3.2.1.3.10 Test passati richiesti** Lo strumento scelto per il calcolo del valore di questa metrica è Tracy$_g$, il quale permette di tracciare i requisiti e mapparli su use case, moduli e classi.

**3.2.1.3.11 Failure avoidance** Lo strumento scelto per il calcolo del valore di questa metrica è il Framework$_g$ Espresso$_g$, fornito dalla Android$_g$ Testing Support Library.

**3.2.1.3.12 Breakdown avoidance** Lo strumento scelto per il calcolo del valore di questa metrica è il Framework$_g$ Espresso$_g$, fornito dalla Android$_g$ Testing Support Library.

### 3.2.2 Obiettivi

**3.2.2.1 Classificazione degli obiettivi** Per garantire la qualità del lavoro del team$_g$, gli *Amministratori* hanno definito degli obiettivi di qualità, riportandole nel *Piano di qualifica*, che devono rispettare la seguente notazione:

**OQ[X][Y][Z]**

dove:

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

Riassumendo, le possibiltà per le denominazione di una metrica sono le seguenti:

- OQPC[Z];
- OQPRD[Z];
- OQPRS[Z].

## 3.3 Configurazione

### 3.3.1 Controllo di versione

Il controllo di versione di documenti e file sorgente viene fatto utilizzando il software$_g$ Git$_g$ e la piattaforma GitHub$_g$. Ad ogni modifica sostanziale ad un documento o ad un file sorgente impone che a quest'ultimo venga assegnato nuovo numero di versione.

Per quanto riguarda le norme di versionamento dei file inerenti alla documentazione si rimanda alla sezione Versionamento dei documenti del presente documento. Per quanto riguarda le norme di versionamento dei file inerenti al codice si rimanda alla sezione Versionamento dei file di codice del presente documento.

**3.3.1.1 Richieste di modifica** Ogni componente del gruppo può avanzare una richiesta di modifica al *Responsabile di progetto*, qualora lo ritenga necessario. Il *Responsabile di progetto* ha il compito di analizzare tale richiesta e decidere se approvarla o meno. In caso affermativo, deve assegnare il compito di realizzare tale modifica ad un membro del gruppo, in base al ruolo da lui ricoperto in quel momento. Una volta effettuata la modifica questa deve essere sottoposta a verifica e dev'essere fatta mantenendo traccia dello stato precedente. In ogni caso, sia di accettazione della richiesta di modifica, sia in caso di rifiuto, il *Responsabile di progetto* è tenuto a motivare la scelta.

**3.3.1.2 Repository** Per organizzare e gestire meglio i file, nonché la dimensione del repository in locale, i file prodotti dal team si dividono in due categorie: documenti e codice sorgente, quindi sono mantenuti in due repository$_g$ distinti.

**Documents:** comprende i file necessari a redigere i documenti e gli strumenti che agiscono su di essi;

**clips:** comprende i file necessari alla creazione del prodotto software.

**3.3.1.2.1 Struttura del repository Documents** I file all'interno del repository$_g$ Documents verranno organizzati secondo questa struttura:

- /Documents
  - /ExternalDocuments
    - /AnalisiDeiRequisiti
    - /DefinizioneDiProdotto

        * /Glossario
        * /ManualeUtente
        * /ManualeSviluppatore
        * /PianoDiProgetto
        * /PianoDiQualifica
        * /VerbaliEsterni
    - /InternalDocuments
        * /NormeDiProgetto
        * /StudioDiFattibilità
        * /VerbaliInterni
- /Tools

#### 3.3.1.2.2 Struttura del repository clips

- /app
    - /src
        * /androidTest
        * /main
        * /test
- /gradle

**3.3.1.2.3 Commit** L'esecuzione di ogni comando commit deve sottostare alle seguenti norme:

- ad ogni commit è necessario specificare un messaggio nel quale si deve dare una descrizione sintetica e più possibile precisa delle modifiche effettuate. Dove possibile, indicare anche l'identificatore univoco della issue$_g$ a cui si riferisce il commit;
- le modifiche apportate devono essere complete e testate con successo;
- è vietato l'utilizzo del comando `git add *` prima di un comando commit per evitare di includere nel repository$_g$ file nascosti, temporanei o non voluti;
- è consigliato specificare il riassunto delle modifiche apportate ad ogni commit tramite il comando `git commit` rispetto al comando `git commit -m`, dove il messaggio deve essere scritto inline tra virgolette.

**3.3.1.2.4 Visibilità del repository** Per la condivisione e il versionamento dei configuration item sono stati creati due repository$_g$ su GitHub$_g$ all'interno del gruppo di lavoro *LeafSWE* composto dai membri del gruppo *Leaf*, accessibile all'indirizzo https://github.com/LeafSWE.

**3.3.1.2.5 File**

**3.3.1.2.5.1 Nomi dei file** I nomi dei file interni al repository$_g$ devono sottostare alle seguenti norme:

- devono contenere solo lettere, numeri, il carattere '_', il segno meno ed il punto;
- devono avere lunghezza minima di tre caratteri;
- devono identificare in modo non ambiguo i file;
- devono riportare le informazioni dal generale al particolare;
- devono, nel caso contengano date, rispettare il formato YYYY-MM-DD.

È consigliato invece:

- utilizzare la notazione CamelCase$_g$, invece di caratteri '_' e '-' qualora il nome di un file fosse composto da più parole;
- utilizzare nomi né troppo lunghi, né troppo corti, indicativamente tra i 10 e i 25 caratteri, estensione compresa;
- specificare, quando possibile, l'estensione del file.

**3.3.1.2.5.2 Codifica dei file** I file contenenti codice oppure documentazione dovranno utilizzare la codifica UTF-8 senza BOM$_g$.

#### 3.3.1.3 Procedure

**3.3.1.3.1 Procedura di richiesta di modifica** Il membro del gruppo che vuole effettuare una modifica deve presentare una richiesta formale al *Responsabile di progetto*. La richiesta deve avere i seguenti campi:

1. autore: contiene nome, cognome e ruolo del richiedente della modifica;
2. documento: contiene il nome del documento di cui si ritiene andrebbe fatta tale modifica;
3. urgenza: indica una misura di quanto il richiedente ritiene necessario fare una modifica, può assumere solamente tali valori:
   (a) "Alta": in questo caso si ritiene che la modifica da fare sia importante, con pesanti conseguenza nell'organizzazione dello svolgimento delle attività future o che possa in qualche modo compromettere l'organizzazione delle attività future;
   (b) "Media": in questo caso il richiedente considera la modifica importante, ma non comporta forti conseguenza nell'organizzazione generale delle attività, ma può influenzare lo svolgimento di alcune di queste;
   (c) "Bassa": in questo caso la modifica proposta di modifica è di importanza secondaria, un suggerimento nell'effettuare alcune attività oppure che non influisce, o il suo peso è poco rilevante, nello svolgimento delle attività.
4. descrizione: una descrizione dettagliata e motivata delle modifiche che si vogliono apportare;
5. decisione del *Responsabile di progetto*. Questo campo va aggiunto successivamente alla decisione del *Responsabile di progetto*. Questo campo può assumere due valori:
   (a) "Approvata";
   (b) "Respinta".

   In caso lo ritenga necessario, il *Responsabile di progetto* può aggiungere a questo campo la motivazione per cui ha approvato o meno la modifica.

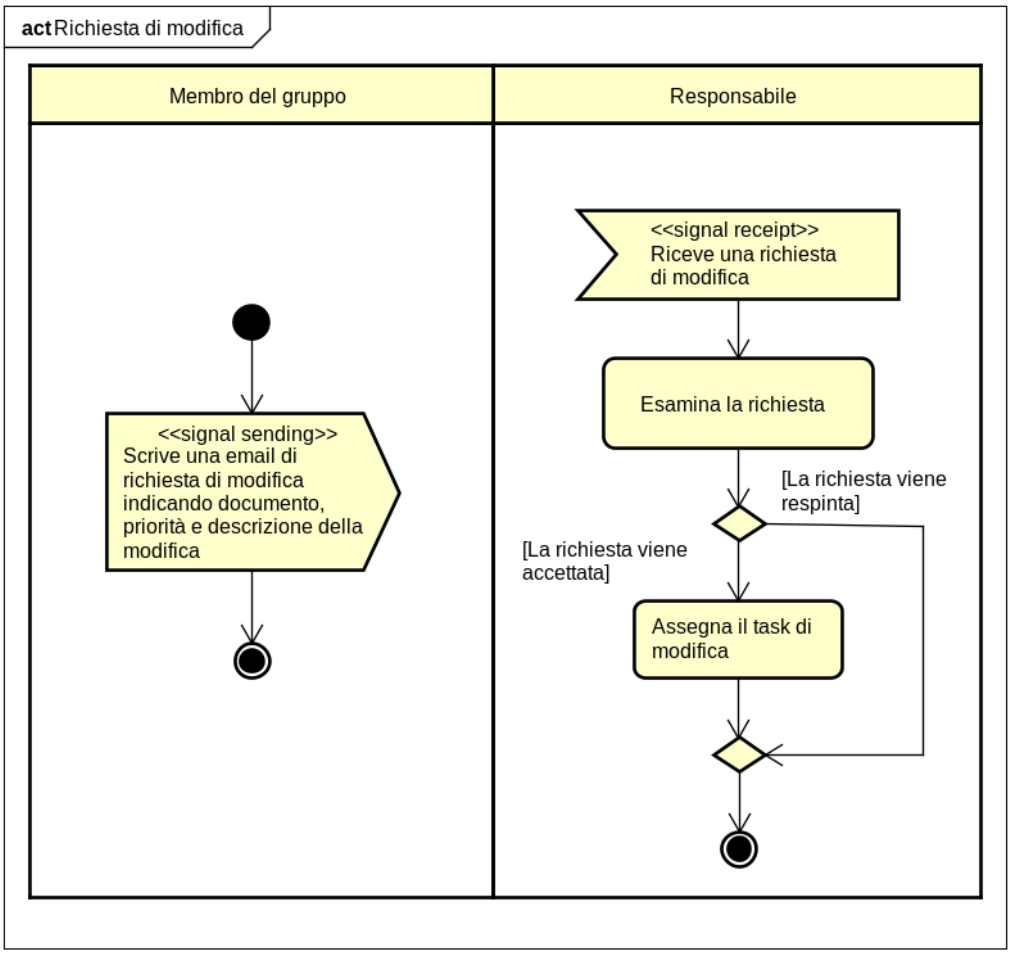


**Figura 2:** Procedura di richiesta modifica di un documento

**3.3.1.3.2 Aggiornamento del repository Documents** Per l'aggiornamento del repository$_g$ Documents è prevista la seguente procedura:

- dare il comando `git pull`. Nel caso in cui si verifichino dei conflitti:
  - dare il comando `git stash` per accantonare momentaneamente le modifiche apportate;
  - dare il comando `git pull`;
  - dare il comando `git stash apply` per ripristinare le modifiche.

  In questo modo il repository$_g$ risulta aggiornato rispetto il repository$_g$ remoto;

- dare il comando `git add NomeDelFile`, dove al posto di "NomeDelFile" si deve mettere il nome del file, o lista di nomi dei file, sul quale sono state effettuate delle modifiche;
- dare il comando `git commit`, e successivamente specificare sinteticamente il riassunto delle modifiche effettuate;
- dare il comando `git push`.

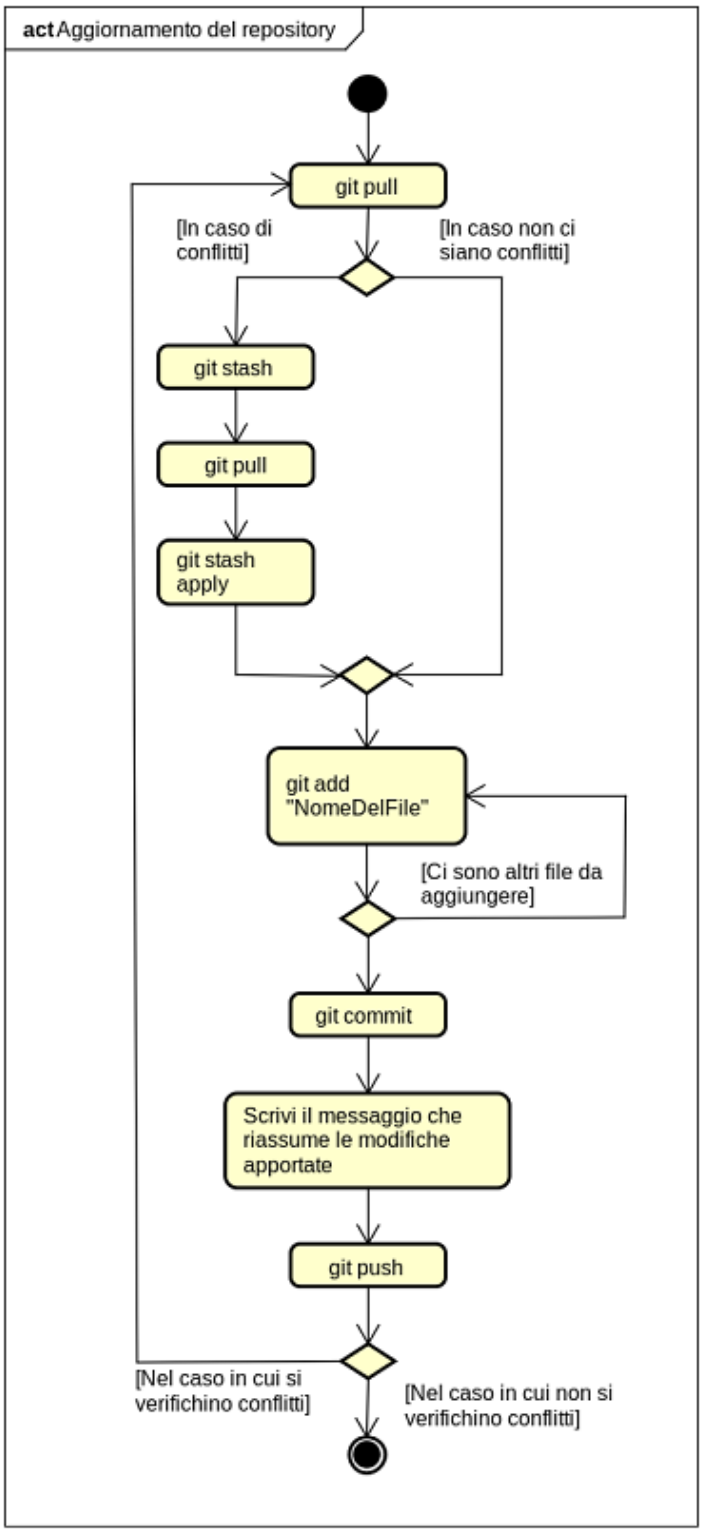


**Figura 3:** Procedura di aggiornamento del repository Documents

**3.3.1.3.3 Aggiornamento del repository clips** Per l'aggiornamento del repository$_g$ clips è prevista la seguente procedura:

1. Usare il comando `git checkout type/nome_feature development`, sostituendo con `type` la tipologia di branch:
   - `f` (feature) per una nuova funzionalità;
   - `d` (debug) per eseguire test e verifiche per isolare e riprodurre la procedura che causa un bug;
   - `t` (test) per sviluppare test per il prototipo;

   e con `nome_feature` il nome della caratteristica da aggiungere che diventerà il nome del nuovo branch;
2. Eseguire le modifiche necessarie nel branch appena creato per l'aggiunta della nuova caratteristica usando i comandi:
   - `git add NomeFile`;
   - `git commit`;
   - `git push`;
3. Una volta che la modifica è stata completata e testata aprire la pagina github del repository ed eseguire una pull request specificando il branch `development` come *base* e il branch `nome_feature` come *compare*. Infine selezionare il pulsante *'Create pull request'*;
4. Successivamente alla *pull request* vengono eseguiti con lo strumento Travis CI$_g$ tutti i test, se il risultato è positivo il *Responsabile di progetto*è incaricato di accettare la *pull request* e chiudere così il branch `nome_feature` altrimenti la *pull request* non è accettata e si torna al punto 2.

**3.3.1.3.4 Esecuzione di un commit** Per facilitare la stesura del diario delle modifiche, sono stati aggiunti al repository$_g$ degli hook, che permettono di effettuare operazioni prima e/o dopo l'esecuzione di un comando Git$_g$. La procedura ridefinita per l'esecuzione di un commit viene descritta dal seguente diagramma.

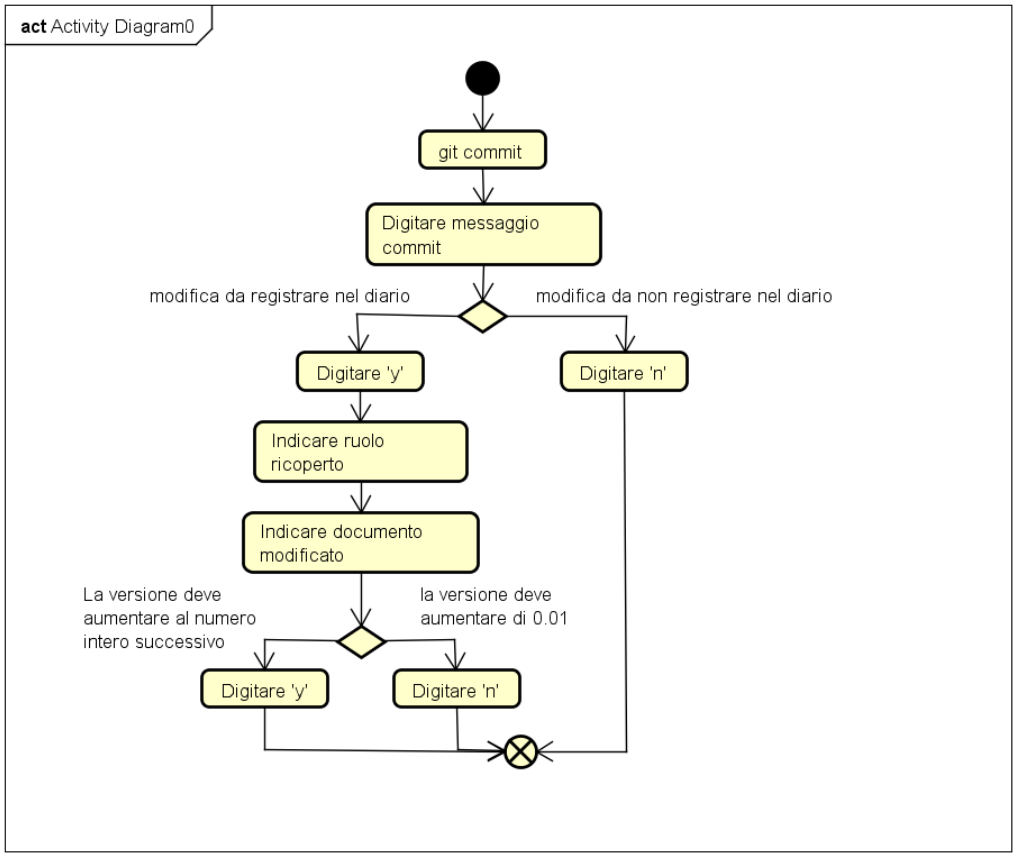


**Figura 4:** Procedura di commit sul repository

**3.3.1.4 Strumenti** Lo strumento utilizzato per il versionamento di documenti e file sorgenti è Git$_g$. Nonostante siano presenti varie alternative, come Mercurial, Subversion e Bazaar, è stato scelto Git$_g$ in quanto soddisfa appieno le necessità di versionamento dei file per questo progetto e, inoltre, permette di versionare i file localmente, senza bisogno di una connessione internet attiva. Come servizio di hosting per il repository$_g$ è stato scelto, invece, GitHub$_g$. Entrambe le scelte sono state fatte anche perché più membri del gruppo avevano già avuto la possibilità di lavorare con tali servizi. Lo strumento utilizzato per effettuare richieste di modifica è Notebook$_g$messo a disposizione da Teamwork$_g$.

## 3.4 Verifica

### 3.4.1 Documenti

Il *Responsabile di progetto* ha il compito di dare inizio alla fase$_g$ di verifica, assegnando i compiti ai *Verificatori*. Quest'ultimi, devono effettuare un'accurata verifica delle seguenti regole:

- deve essere utilizzata una sintassi corretta e il più possibile semplice;
- devono essere utilizzati periodi brevi;
- la struttura del documento deve essere semplice e intuitiva;
- devono essere rispettate le norme tipografiche;
- devono essere rispettate le regole riguardanti il glossario.

Durante l'intera attività di verifica i *Verificatori* dovranno utilizzare lo strumento del diario delle modifiche in modo tale da concentrare la loro attenzione sulle modifiche effettuate ad un documento e tenere traccia degli errori più comuni commessi nel redigere i documenti.

**3.4.1.1 Sintassi** I *Verificatori* hanno il compito di identificare errori sintattici nei documenti. Ciò può essere fatto con l'ausilio di strumenti automatici, ma ogni documento deve essere sempre sottoposto a walkthrough in modo tale da individuare errori che non sono stati in grado di evidenziare. Inoltre hanno il compito di inserire nel *Glossario* tutte quelle parole che possono essere fonte di ambiguità.

**3.4.1.2 Periodi** I *Verificatori* hanno il compito di calcolare l'indice Gulpease$_g$ di ogni documento utilizzando lo strumento automatico preposto (vedi sezione Strumenti). Qualora non si ottenga un risultato soddisfacente il *Verificatore* deve applicare la tecnica walkthrough con l'obiettivo di individuare periodi troppo lunghi, che possono essere di difficile comprensione o leggibilità.

**3.4.1.3 Struttura del documento** I *Verificatori* devono verificare che la struttura di ogni documento rispetti i contenuti del documento stesso.

**3.4.1.4 Procedure**

**3.4.1.4.1 Resoconto attività di verifica sui documenti** La stesura dell'attività del resoconto dell'attività di verifica dovrà essere fatta al termine dell'attività di verifica. Il *Verificatore* dovrà riportare il numero di occorrenze per ognuna delle voci definite nella sezione norme. Qualora fossero stati trovati degli errori nelle attività di verifica precedenti, il numero di errori trovati andrà a sommarsi al numero di errori trovati nelle attività precedenti. Inoltre, il *Verificatore*, nel caso in cui dovesse trovare termini mancanti nel *Glossario*, dovrà aggiornare la tabella definita nelle norme che regolano l'attività manuale di verifica.

**3.4.1.4.1.1 Attività manuale di verifica** Ogni attività manuale di verifica sui documenti deve essere accompagnata da un resoconto. Il resoconto deve contenere il numero di occorrenze riscontrate durante la verifica per ognuna delle seguenti voci:

- periodi troppo lunghi o complessi;
- parole non appropriate;
- incongruenze;
- errori concettuali;
- violazioni di quanto stabilito nelle norme tipografiche.

Al fine di rendere più veloce e chiara l'analisi di queste voci, è consigliabile che questi dati vengano rappresentati in forma tabellare.
Inoltre, deve essere sempre presente una tabella contenente il numero di occorrenze riscontrate durante la verifica per ognuna delle seguenti voci:

- termini candidati ad essere aggiunti al *Glossario*;
- termini aggiunti al *Glossario*.

**3.4.1.4.1.2 Attività automatica di verifica** Ogni attività automatica di verifica sui documenti deve essere accompagnata da un resoconto. Il resoconto deve contenere il numero di occorrenze riscontrate durante la verifica per ognuna delle seguenti voci:

- errori ortografici;
- utilizzo errato LaTeX$_g$.

Inoltre, devono essere riportarti gli indici Gulpease calcolati per ognuno dei documenti sui quali è stata effettuata attività di verifica.

### 3.4.2 Diagrammi UML

I *Verificatori* devono controllare tutti i diagrammi $UML_g$ prodotti, sia che venga rispettato lo standard $UML_g$, sia che siano corretti semanticamente. I diagrammi utilizzati finora sono solamente i diagrammi dei casi d'uso.

**3.4.2.1 Diagrammi dei casi d'uso** Le verifiche che devono essere effettuate ai diagrammi dei casi d'uso, innanzitutto, devono riguardare il rispetto dello standard $UML_g$, soprattutto riguardanti inclusioni, generalizzazioni e estensioni. Successivamente si deve verificare che la semantica del diagramma sia corretta, ovvero che il diagramma rappresenti effettivamente ciò che si vorrebbe descrivere e modellare. Per quest'ultimo punto bisogna porre particolare attenzione sull'identificazione degli attori. Per questo motivo è prevista una procedura di verifica degli attori dei diagrammi UML in questo documento.

#### 3.4.2.2 Procedure

**3.4.2.2.1 Procedura di verifica attori dei diagrammi UML** Per identificare un attore può risultare utile seguire la seguente procedura:

1. per prima cosa è utile chiedersi se ciò che vogliamo rappresentare come attore è una persona che interagisce col sistema. In caso affermativo è giusto rappresentarlo come attore, in caso negativo si deve andare al passo 2;

2. se ciò che vogliamo rappresentare come attore non è una persona che interagisce col sistema bisogna chiedersi se questa "cosa" può cambiare insieme al design del sistema. In caso affermativo probabilmente ciò che vogliamo rappresentare non è un attore ma una parte del sistema stesso. In caso negativo, invece, è con buona probabilità un attore.

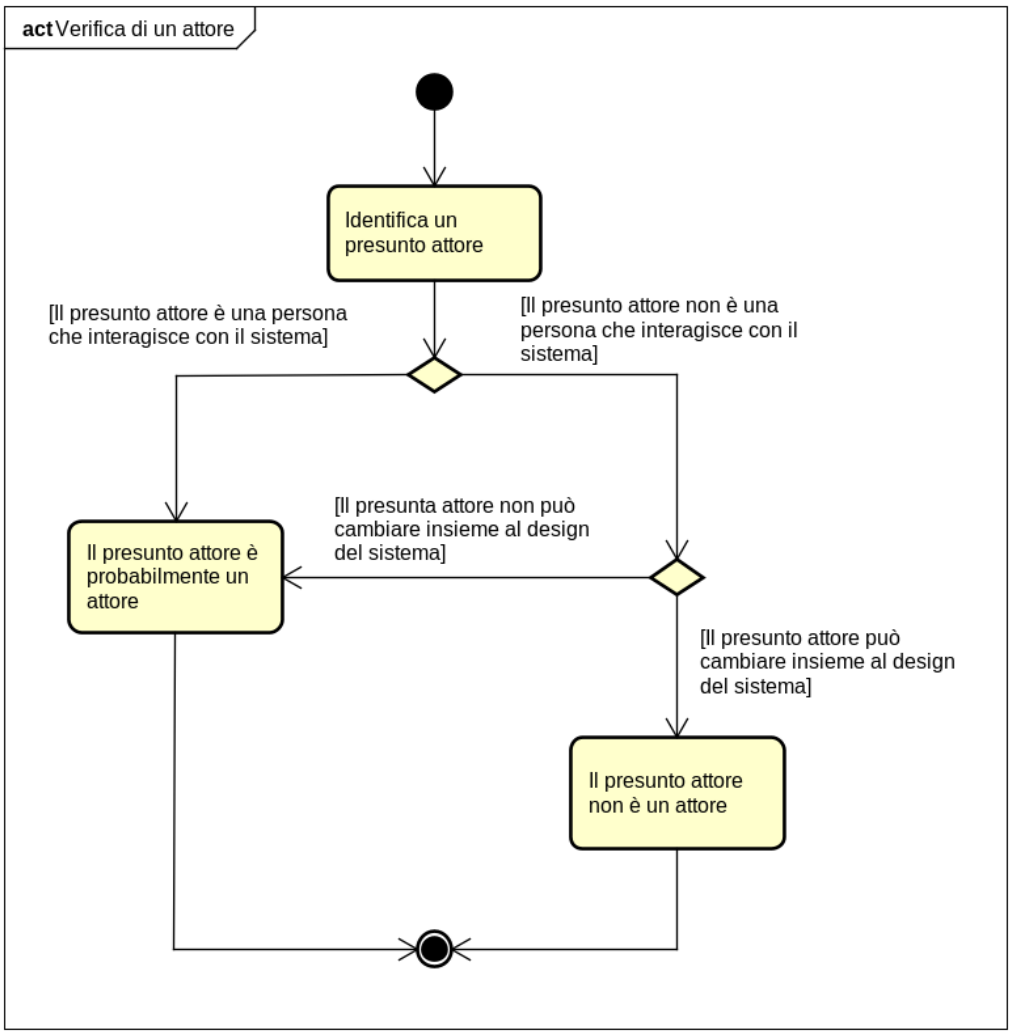


**Figura 5:** Procedura di verifica di un attore

### 3.4.3 Issue tracking

L'issue$_g$ tracking è un'attività di supporto per i *Verificatori* ai quali permette di tenere traccia potenziali errori, segnalandoli al *Responsabile di progetto*. Lo strumento scelto per la gestione di questa attività è GitHub(vedi sezione Strumenti). Questo strumento è utile anche al *Responsabile di progetto* per assegnare i compiti di correzione degli errori.

#### 3.4.3.1 Sintassi di una label

Le label sono delle etichette che è possibile associare ad ogni issue$_g$. Ogni etichetta rappresenta lo stato nel quale si trova la issue$_g$. Le uniche etichette che sono accettate sono le seguenti:

- **Request**: rappresenta la richiesta di una issue$_g$;
- **Question**: identifica una issue$_g$ nella quale si sta discutendo per risolvere il problema;
- **ToDo**: identifica una issue$_g$ per la quale è stata trovata la soluzione ma questa deve ancora essere applicata;
- **Working**: identifica una issue$_g$ alla quale un membro del gruppo sta lavorando per applicare la soluzione;
- **Rejected**: identifica una issue$_g$ che non è stata accettata poiché risulta essere futile o dannosa;
- **ToVerify**: identifica una issue$_g$ che è stata completata e deve essere verificata.

**3.4.3.2 Sintassi di una Issue** Ogni issue$_g$ avrà un nome che dovrà seguire la seguente notazione:

**[D]:[S]**

dove:

- **D** rappresenta la sigla del documento di interesse;
- **S** rappresenta la sintesi della descrizione del problema.

Segue, poi, la descrizione della issue$_g$. Questa dovrà obbligatoriamente contenere:

- dove si trova il problema. Questo va specificato tramite sezione e, possibilmente, numero di riga dove inizia il problema;
- una descrizione dettagliata del problema che deve comprendere:
  - tutte gli errori associati al problema. Questi devono essere riportati nella forma di checkbox. Per fare ciò è necessario premettere alla descrizione di ogni errore i caratteri `- [ ]`. Questo permette di facilitare il tracciamento dello stato di avanzamento della issue$_g$.
  - una eventuale porzione di codice che potrebbe essere causa del problema. Per fare ciò è consigliabile utilizzare il comando:

````
```nomelinguaggio
codice da riportare
``` .
````

Sostituendo “nomelinguaggio” al nome del linguaggio di programmazione utilizzato, si potrà vedere la porzione di codice evidenziata secondo la sintassi del linguaggio stesso, facilitandone la lettura.

- la motivazione per cui si è ritenuto necessario sollevare una issue$_g$.

#### 3.4.3.3 Procedure

**3.4.3.3.1 Gestione di una issue** Ai *Verificatori* viene chiesto di verificare il lavoro fatto da un altro membro del gruppo riportato nel dettaglio in una issue$_g$.
Qualora i *Verificatori* dovessero riscontrare delle anomalie, la procedura da seguire è la seguente:

1. il *Verificatore* che ha riscontrato delle anomalie dovrà segnalarlo commentando la issue$_g$ che ha verificato e modificando la label da "ToVerify" a "ToDo". Nel caso in cui non vengano riscontrare anomalie, può chiudere l'issue$_g$ aggiungendo al messaggio di commit la stringa "close `#numeroIssue`";

2. l'assegnatario della issue$_g$ dovrà provvedere ad apportare le modifiche rilevate dal *Verificatore* nel caso le ritenesse necessarie;

3. il *Verificatore* dovrà procedere ad una nuova verifica, ritornando al punto 1.

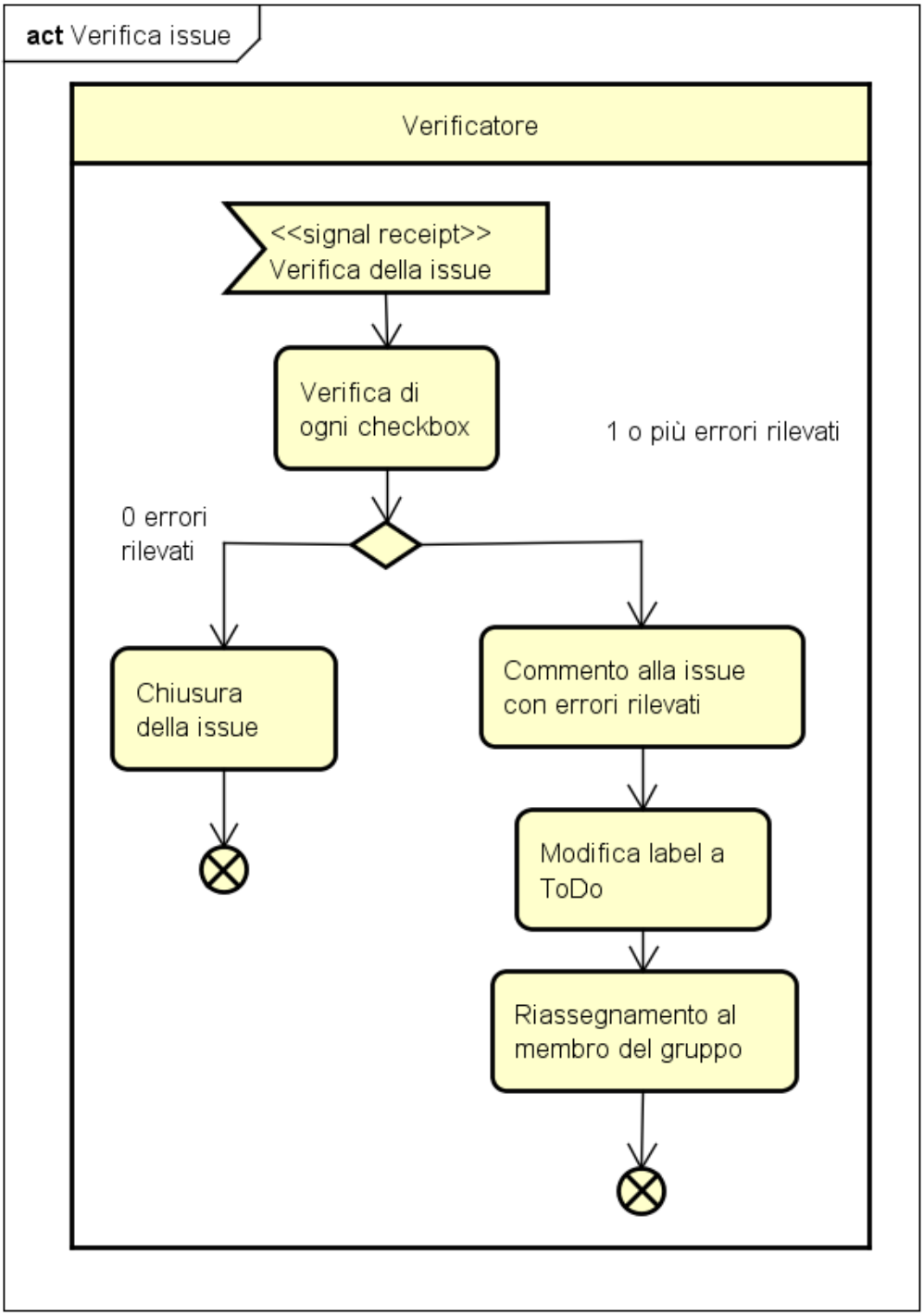


**Figura 6:** Procedura di gestione di una issue già aperta

Nel caso in cui i *Verificatori* dovessero riscontrare delle anomalie che non riguardano una issue$_g$ specifica, la procedura da seguire è la seguente:

1. il *Verificatore* che ha riscontrato un'anomalia$_g$ dovrà aprire una nuova issue$_g$ assegnandole l'etichetta Request;

2. il *Responsabile di progetto* dovrà valutare la issue$_g$ e decidere se:

   - la issue$_g$ risulta essere futile o dannosa. In questo caso verrà rimossa l'etichetta Request ed assegnata l'etichetta Rejected. La issue$_g$ verrà poi chiusa, terminando la procedura.
   - la issue$_g$ risulta essere appropriata. In questo caso verrà rimossa l'etichetta Request ed assegnata l'etichetta Question.

3. i membri del gruppo dovranno discutere e proporre nuove idee al fine di risolvere la issue$_g$. Questa discussione non dovrà protrarsi per più di tre giorni. Qualora non si dovesse essere trovata una soluzione ottimale, il *Responsabile di progetto* dovrà decidere quale sia la soluzione migliore.

4. il *Responsabile di progetto*, trovata la soluzione da applicare, assegnerà, al membro del gruppo da lui ritenuto opportuno, un ticket$_g$. L'assegnatario del ticket$_g$ è la persona incaricata all'applicazione della soluzione relativa alla issue$_g$. Inoltre, il *Responsabile di progetto* dovrà rimuovere dalla issue$_g$ l'etichetta Question ed assegnare l'etichetta ToDo.

5. quando l'incaricato inizierà a lavorare sulla issue$_g$ rimuoverà l'etichetta ToDo ed assegnerà l'etichetta Working. Ad ogni nuovo commit, l'incaricato dovrà riferire, nel messaggio del commit stesso, la issue$_g$ sulla quale sta lavorando. Questo permetterà di tener traccia delle modifiche effettuate per risolvere la issue$_g$.

6. l'incaricato dovrà inserire nel messaggio del suo ultimo commit di modifica la stringa `£toVerify`, in modo che la label della issue$_g$ venga modificata da Working a ToVerify. A questo punto, i *Verificatori* potranno verificare il lavoro svolto.

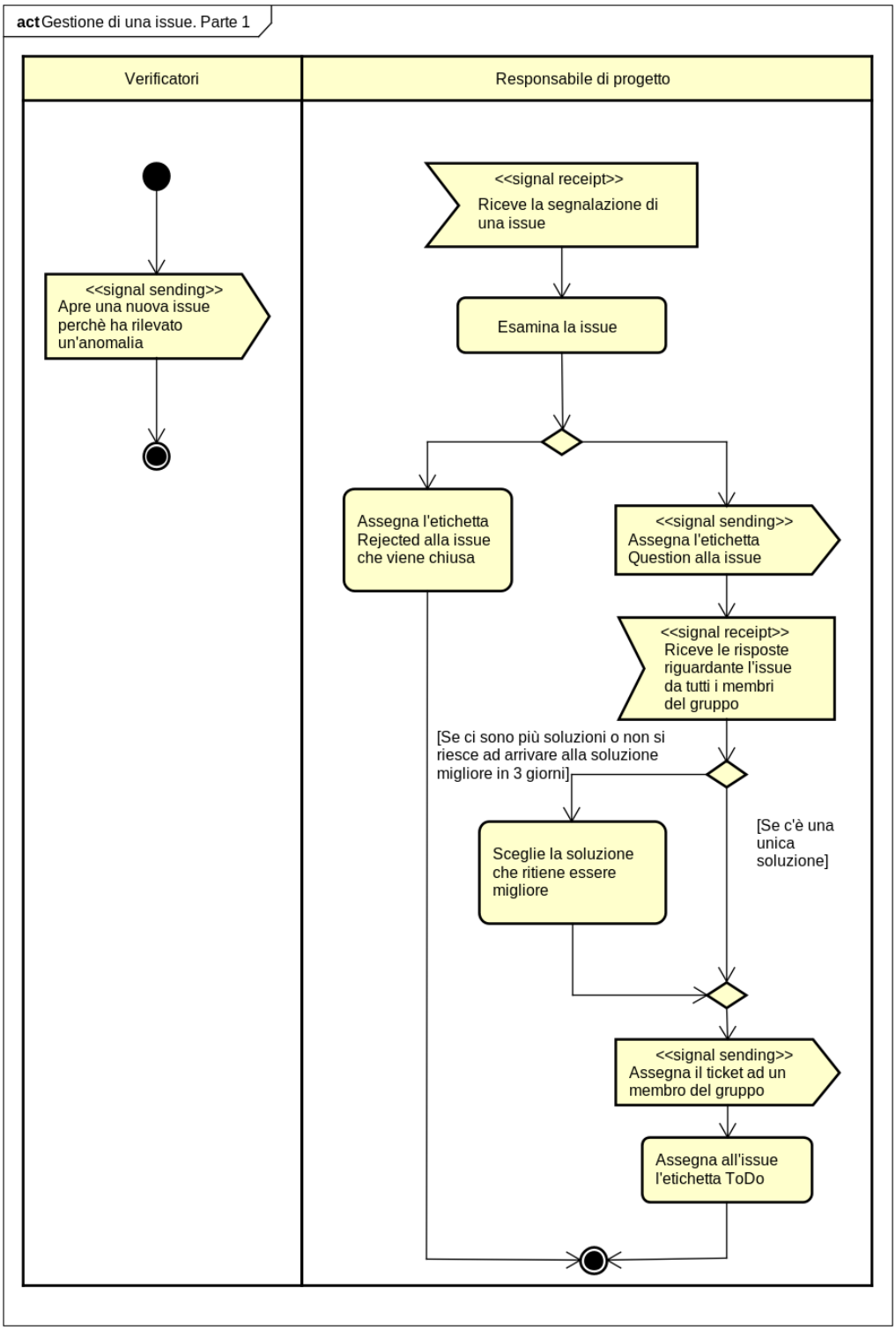


**Figura 7:** Procedura di gestione di una issue. Parte 1

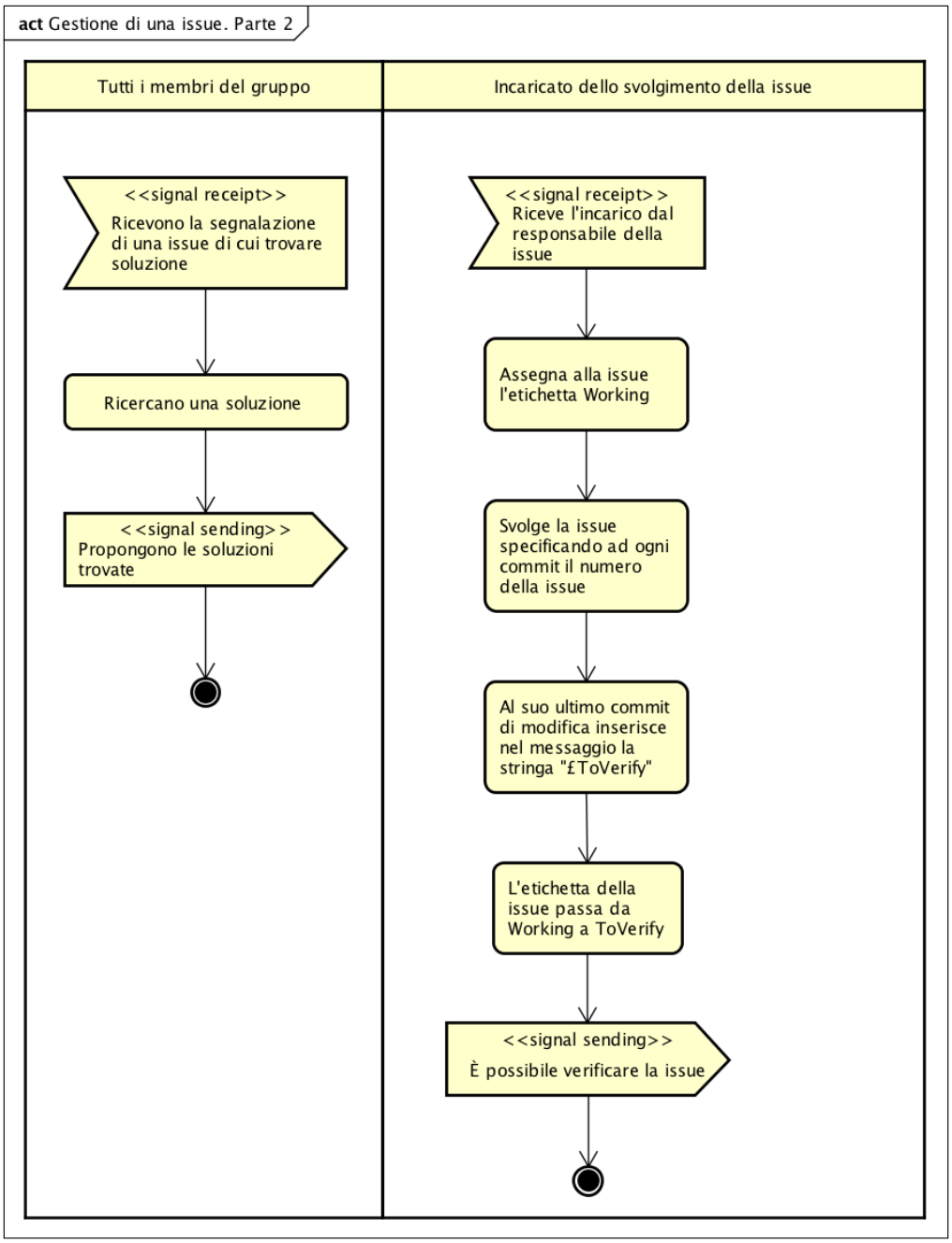


**Figura 8:** Procedura di gestione di una issue. Parte 2

#### 3.4.3.4 Strumenti

**3.4.3.4.1 Strumento per l'issue tracking** Lo strumento scelto per fare il tracking delle issue$_g$ è il servizio Issues messo a disposizione da GitHub$_g$.

### 3.4.4 Analisi

Per poter verificare la qualità del prodotto$_g$ è stato scelto di applicare delle tecniche di analisi sui documenti e sul codice.

**3.4.4.1 Analisi statica** Per analisi statica si intende la valutazione di un sistema basata su struttura, contenuto e documentazione senza che questo venga eseguito. Questa tecnica è applicabile sia al codice, che alla documentazione stessa. L'analisi statica può avvenire con due modalità:

- **walkthrough**: questa tecnica dev'essere applicata quando non si sa che errori o problematiche si stanno cercando. La tecnica consiste nel leggere il codice sorgente o il documento da cima in fondo per trovare anomalie di qualsiasi tipo;
- **inspection**: questa tecnica dev'essere applicata quando si ha idea della problematica che si sta cercando e consiste in una lettura mirata del documento/codice, sulla base di una lista degli errori precedentemente stilata.

La tecnica walkthrough è molto onerosa e deve essere utilizzata soprattutto nelle prime fasi del progetto, quando non è già presente una lista degli errori comuni, oppure non si è ancora sufficientemente preparati riguardo un aspetto del progetto. Avanzando nelle fasi del progetto sarà utile stilare una lista quanto più possibile completa di errori comuni, in modo tale da evitare la tecnica walkthrough e applicare l'inspection.

**3.4.4.2 Analisi dinamica** L'analisi dinamica è una forma di valutazione di un sistema software$_g$, oppure di qualche suo componente, basato sull'osservazione del suo comportamento durante l'esecuzione. Questa tecnica non è applicabile, quindi, per trovare errori nella documentazione. I test che devono essere implementati devono essere in numero relativamente ridotto e il più possibile di valore dimostrativo.

## 3.5 Validazione

Il processo$_g$ di validazione ha l'obbiettivo di verificare che:

- il prodotto$_g$ finale sia conforme a quanto è stato pianificato;
- il prodotto$_g$ finale sia abile nell'isolare e minimizzare gli effetti degli errori.

### 3.5.1 Responsabilità

Di seguito si elencano le rispettive responsabilità per la validazione del prodotto$_g$:

- i *Verificatori* hanno il compito di eseguire con attenzione i test, tracciandone i risultati che andranno poi valutati;
- il *Responsabile di progetto* revisiona i risultati dei test e decide se accettarli o ripeterli. Ha inoltre il compito di informare il committente dell'esecuzione e risultato dei test, fornendo indicazioni sulla possibilità di eseguirli in modo indipendente.

### 3.5.2 Procedure

**3.5.2.1 Procedura per la validazione** I passi per eseguire l'attività di validazione sono i seguenti:

1. i *Verificatori* eseguono manualmente i test sul prodotto$_g$ finale, tracciando accuratamente i risultati ottenuti;
2. il *Responsabile di progetto* di progetto analizza i risultati e decide se:
   (a) accettarli;
   (b) chiedere una ripetizione di alcuni o tutti i test, eventualmente con verificatori diversi.
3. una volta accettati i test, il *Responsabile di progetto* consegna i risultati al proponente, informandolo sulle modalità di esecuzione indipendente della validazione.

# 4 Processi organizzativi

## 4.1 Gestione organizzativa

### 4.1.1 Comunicazione

**4.1.1.1 Comunicazioni interne** Per le comunicazioni interne è stato scelto di utilizzare lo strumento Messages messo a disposizione da Teamwork$_g$. La procedura per inviare un messaggio ai membri del gruppo è la seguente:

1. accedere a Teamwork$_g$ ed entrare nella sezione Messages;
2. creare un nuovo messaggio inserendo oggetto e corpo del messaggio;
3. selezionare i destinatari del messaggio ed inviare il messaggio.

Per rispondere ad un messaggio ricevuto la procedura è la seguente:

1. accedere a Teamwork$_g$ ed entrare nella sezione Messages;
2. selezionare il messaggio al quale si vuole rispondere ed inviare la risposta.

Tale sistema deve essere utilizzato solo per questioni riguardanti il progetto. Per i messaggi brevi, al fine di velocizzare le comunicazioni interne, il gruppo utilizzerà lo strumento di messaggistica Telegram$_g$. Inoltre, per le videochiamate, è stato scelto di utilizzare Skype$_g$.
Qualora i sistemi sopra elencati venissero utilizzati per decisioni rilevanti per lo sviluppo del progetto è necessario stilare un verbale.
I membri del gruppo sono tenuti a prestare attenzione al numero di messaggi diffusi per non creare difficoltà di comunicazione.

**4.1.1.2 Comunicazioni esterne** Per la comunicazioni esterne è stato creato un apposito indirizzo di posta elettronica: `leaf.gruppo@gmail.com`. Il *Responsabile di progetto* ha l'incarico di mantenere i contatti tra il team$_g$ e le componenti esterne utilizzando tale indirizzo di posta elettronica. Inoltre è suo compito informare i membri del gruppo delle discussioni avvenute con componenti esterne: questo può essere fatto riassumendo la conversazione in una email e inviandola alla mailing list del gruppo.

**4.1.1.3 Composizione delle email** Questa sezione tratta le norme da rispettare nella composizione delle email, sia per la comunicazione interna che esterna.

**4.1.1.3.1 Mittente** Nel caso di comunicazione interna il mittente dovrà essere l'indirizzo di posta elettronica personale del membro del gruppo che ha scritto l'email, mentre in caso di comunicazione esterna l'unico indirizzo che deve essere utilizzato è leaf.gruppo@gmail.com.
Qualora la comunicazione debba avvenire tra un numero ristretto di persone all'interno del gruppo, questi potranno utilizzare i loro indirizzi email personali.

**4.1.1.3.2 Destinatario** Nel caso di comunicazione interna al gruppo l'unico destinatario deve essere leaf.gruppo@gmail.com per facilitare l'invio a tutti i membri del gruppo stesso, mentre in caso di comunicazione esterna il destinatari possono essere il Prof. Tullio Vardanega, il Prof. Riccardo Cardin oppure i proponenti del progetto, a seconda dello scopo dell'email.

**4.1.1.3.3 Oggetto** L'oggetto dell'email deve sintetizzare il contenuto dell'email stessa in modo più chiaro ed esaustivo possibile. Possibilmente l'oggetto di una nuova email deve essere differente rispetto alle email inviate e ricevute in precedenza, in modo tale da rendere facilmente identificabile ogni messaggio.
Per comporre un messaggio di risposta è necessario anteporre all'oggetto il prefisso "Re:", mentre nel caso di inoltro di un messaggio è obbligatorio aggiungere il prefisso "I:". In entrambi i casi, le rimanenti parti dell'email non vanno modificate.

**4.1.1.3.4 Corpo** Il corpo del messaggio deve essere esaustivo, sintetico e deve essere comprensibile a tutti i destinatari del messaggi.
In caso di risposta o inoltro è preferibile aggiungere la nuova parte di testo all'inizio dell'email per permettere una più facile lettura del contenuto.
Nell'eventualità che nel contenuto di un'email ci si debba riferire a persone è preferibile utilizzare la sintassi: "'Nome Cognome", nel caso invece in cui si debba fare esplicito riferimento ad un ruolo di progetto è consigliabile riportarne per intero il nome.

**4.1.1.3.5 Allegati** Si consiglia di non fare uso di allegati per lo scambio di documenti o file, a meno che non sia strettamente necessario. È preferibile, invece, caricare questi file in una cartella di Google Drive$_g$ e inviare per email il link al documento o file desiderato.
Sia in caso di invio di allegati, che di link a documenti o file è buona norma inserire una breve descrizione di presentazione dell'allegato in modo tale che

sia possibile in modo semplice capire di cosa si tratta.
Infine si chiede di prestare attenzione al formato dei documenti e file.

### 4.1.2 Riunioni

Il *Responsabile di progetto* ha il compito di indire le riunioni sia interne che esterne. Per ogni riunione il *Responsabile di progetto* dovrà inviare un messaggio tramite l'apposito strumento messo a disposizione da Teamwork$_g$, con la seguente struttura:

- **oggetto**: convocazione della riunione N per il giorno AAAA-mm-GG;
- **corpo**:
  - data;
  - ora;
  - luogo;
  - tipo di riunione;
  - ordine del giorno.

Dove N rappresenta il numero della riunione e tipo indica se la riunione sia interna richiesta dal *Responsabile di progetto*, interna richiesta da uno o più dai membri del gruppo oppure esterna.
Le informazioni sulle riunioni devono essere presentate con più preavviso possibile, almeno tre giorni prima in modo tale che i membri del gruppo possano organizzare i loro impegni ed essere presenti alla riunione.

#### 4.1.2.1 Riunioni interne

**4.1.2.1.1 Convocazione di una riunione interna** In generale, il compito di convocare le riunioni interne spetta al *Responsabile di progetto*, che può indire le riunioni quando più lo ritiene opportuno. Gli altri componenti del gruppo possono richiedere una riunione interna straordinaria, presentando al *Responsabile di progetto* le motivazioni per le quali si ritiene necessaria una riunione. In questi casi il *Responsabile di progetto* può:

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

In ogni caso spetta al *Responsabile di progetto* decidere data, ora e luogo dell'incontro contattando i membri del team$_g$ e chiedendo loro la disponibilità. Questi sono tenuti a rispondere tempestivamente in modo tale da dare la possibilità al *Responsabile di progetto* di anticipare o posticipare la data o l'ora della riunione.

**4.1.2.1.2 Gestione di una riunione interna** All'inizio di ogni riunione interna il *Responsabile di progetto* nomina un segretario che ha il compito di redigere la minuta della riunione, catturando possibilmente tutti e soli gli aspetti più importanti della riunione stessa. Terminato l'incontro, il segretario ha il compito di redigere il verbale dell'incontro. Questo verbale verrà archiviato nel repository$_g$ del gruppo, per la consultazione di tutti i membri.
Durante le riunioni i partecipanti devono tenere un comportamento che favorisca la discussione all'interno del gruppo e di tutti gli argomenti previsti.

#### 4.1.2.2 Riunioni esterne

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

**4.1.2.2.2 Gestione di una riunione esterna** Ad ogni riunione esterna il *Responsabile di progetto* deve chiedere la disponibilità ai proponenti o committenti di fare una registrazione audio dell'incontro. Ciò permetterà, alla fine della riunione, di redigere un verbale che sia quanto più fedele possibile a ciò di cui si è discusso durante l'incontro. In questo modo anche eventuali membri assenti alla riunione potranno disporre di un documento che riporti i temi trattati in modo esaustivo. In questo caso, come per le riunioni interne, viene nominato un segretario che ha il compito di riascoltare l'incontro e di scriverne un verbale. In caso non fosse possibile registrare l'incontro il *Responsabile di progetto* deve redigere il verbale della riunione esterna, avvalendosi dell'aiuto di tutti i membri del team$_g$ presenti all'incontro. Questo permetterà di avere una trascrizione più fedele possibile dei contenuti.

**4.1.2.3 Verbale di una riunione** Il verbale di una riunione è un documento nel quale vengono riassunti gli argomenti trattati e nel quale vengono indicate eventuali decisioni prese nel corso della riunione stessa.
Il compito della stesura del verbale spetta:

- ad un segretario, nominato all'inizio di una riunione interna oppure alla fine di una riunione esterna nella quale è stato dato il permesso di registrare l'incontro;
- al *Responsabile di progetto* con l'aiuto di tutto il gruppo, nel caso di riunione esterna senza la possibilità di registrare quanto è stato detto.

I verbali dovranno avere queste struttura:

- dove e quando si è svolta la riunione;
- quali membri hanno partecipato alla riunione e quali membri erano, invece, assenti;
- il nome dell'eventuale segretario oppure del *Responsabile di progetto* che ha redatto il verbale;
- tipo di riunione;
- ordine del giorno;
- eventuali argomenti dell'ordine del giorno non trattati;
- gli interventi più significativi in ordine temporale;
- suggerimenti, proposte, decisioni emerse durante la riunione;
- dubbi, problematiche emerse durante la riunione;
- eventuali compiti assegnati a membri del gruppo, con indicazione di nome, cognome e ruolo della persona a cui è stato assegnato;
- eventuali argomenti da trattare nella prossima riunione.

### 4.1.3 Ruoli di progetto

Per lo sviluppo collaborativo del progetto, ai membri del gruppo saranno assegnati dei ruoli differenti che corrispondono a figure professionali. Ogni membro dovrà ricoprire ogni ruolo almeno una volta ed è necessario garantire che il ruolo di ciascun membro del gruppo non sia in conflitto con il ruolo che ha ricoperto in passato. È compito del *Verificatore* controllare che siano rispettate queste condizioni, in caso contrario dovrà avvertire il *Responsabile di progetto* che dovrà risolvere il problema.

**4.1.3.1 Responsabile di progetto** Il *Responsabile di progetto* rappresenta il progetto presso il fornitore e presso il committente, accentrando su di sé la responsabilità di scelta e approvazione. Per questo motivo ha responsabilità su:

- 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 *Norme di progetto*;
- garantire il rispetto dei ruoli e dei compiti assegnati nel *Piano di progetto*.

**4.1.3.2 Amministratore** L'*Amministratore* è responsabile del controllo e della gestione dell'ambiente di lavoro. In particolare deve preoccuparsi di:

- equipaggiare l'ambiente di lavoro con strumenti, procedure, infrastrutture e servizi a supporto dei processi che permettano di automatizzare il più possibile le attività o parti di esse;
- garantire che l'ambiente di lavoro sia completo, dotato di tutti gli strumenti necessari al progetto, ordinato e aggiornato;
- controllare le versioni e configurazioni del prodotto$_g$;
- gestire la documentazione, controllarne la diffusione, la disponibilità e l'archiviazione;
- fornire procedure e strumenti per il monitoraggio e segnalazione per il controllo qualità;

- risolvere problemi legati alla gestione dei processi tramite l'adozione di strumenti adatti.

L'*Amministratore* redige inoltre le *Norme di progetto* dove viene spiegato l'utilizzo degli strumenti, e deve redigere la sezione del *Piano di qualifica* dove vengono descritti gli strumenti e i metodi atti alla verifica. L'*Amministratore* non compie scelte gestionali, ma tecnologiche concordate con il *Responsabile di progetto*.

**4.1.3.3 Analista** L'*Analista* è il responsabile delle attività di analisi. Le mansioni che gli competono riguardano:

- comprendere la natura e la complessità del problema tramite l'analisi dei bisogni e delle fonti;
- 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*. Partecipa alla definizione del *Piano di qualifica* in quanto conosce a fondo il problema e deve avere chiari i livelli di qualità richiesti, oltre alle procedure per ottenerli.

**4.1.3.4 Progettista** Il *Progettista* è il responsabile delle attività di progettazione. I compiti a lui affidati comprendono:

- produrre una soluzione attuabile e che sia comprensibile e soddisfacente per gli stakeholders;
- effettuare scelte su aspetti progettuali che applichino al prodotto$_g$ soluzioni note ed ottimizzate;
- effettuare scelte su aspetti progettuali e tecnologici che rendano il prodotto$_g$ facilmente manutenibile;
- effettuare scelte su aspetti progettuali e tecnologici che rendano il prodotto$_g$ realizzabile con costi e scadenze prefissate.

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.

**4.1.3.5 Programmatore** Il *Programmatore* è responsabile delle attività di codifica e delle componenti di ausilio necessarie per l'esecuzione delle prove di verifica e validazione. In particolare ha i seguenti compiti:

- implementare in maniera rigorosa le soluzioni descritte dal *Progettista*;
- scrivere codice che sia documentato, manutenibile e che rispetti le metriche stabilite per la scrittura del codice;
- realizzare i test per la verifica e la validazione del codice stesso;
- redigere il manuale per gli sviluppatori.

Il *Programmatore*, infine, deve occuparsi di redigere il *Manuale utente*.

**4.1.3.6 Verificatore** Il *Verificatore* è il responsabile delle attività di verifica. Le mansioni che gli competono sono:

- garantire che l'attuazione delle attività sia conforme alle norme stabilite;
- verificare che le attività svolte non abbiano introdotti errori;
- controllare la conformità di ogni stadio del ciclo di vita del prodotto$_g$.

Il *Verificatore* deve redigere le sezioni del *Piano di qualifica* riguardanti l'esito e la completezza delle verifiche effettuate.

**4.1.3.7 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.

### 4.1.4 Ticketing

**4.1.4.1 Task list** Il *Piano di progetto* prevede la divisione del modello di sviluppo in fasi. L'insieme di attività da svolgere in una certa fase$_g$ è contenuto in una task list$_g$. Il *Responsabile di progetto* deve, per ogni fase$_g$ del modello, creare la relativa task list$_g$ su Teamwork$_g$. Per ogni task list$_g$ valgono le seguenti regole:

- può essere creata solamente dal *Responsabile di progetto*;
- deve avere lo stesso nome della fase$_g$ alla quale si riferisce.

**4.1.4.2 Ticket** I ticket$_g$ rappresentano l'associazione tra un membro del gruppo ed un'attività. Lo strumento scelto per la loro creazione e gestione è Teamwork$_g$(vedi sezione Pianificazione). Per ogni ticket$_g$ valgono le seguenti norme:

- possono essere creati solamente dal *Responsabile di progetto*;
- alla creazione di un nuovo ticket$_g$ l'assegnatario deve esserne informato;
- è consigliabile che non vi siano più di 3 ticket$_g$ per ogni attività. Nel caso in cui questo dovesse accadere, il *Responsabile di progetto* deve valutare la possibilità di suddividere l'attività in sotto-attività più piccole, ciascuna realizzabile da un sola persona.

È consigliabile che i membri del gruppo accedano quotidianamente a Teamwork$_g$ per verificare quali siano i ticket$_g$ a loro assegnati e riportare lo stato di avanzamento dell'attività a loro assegnata.

### 4.1.5 Procedure

**4.1.5.1 Procedura di creazione e gestione di un ticket** Per la creazione di un nuovo ticket$_g$ il *Responsabile di progetto* deve accedere alla sezione Tasks di Teamwork$_g$. La procedura per la creazione di un task$_g$ è la seguente:

1. accedere alla task list$_g$ relativa alla fase$_g$ nella quale si desidera che l'attività sia svolta;
2. creare un nuovo task$_g$;
3. se il task$_g$ viene reputato troppo corposo per essere portato a termine da una singola persona, il *Responsabile di progetto* può suddividerlo in sotto-task$_g$;

4. i task$_g$ e i sotto-task$_g$ vengono assegnati ai membri del gruppo;
5. l'assegnatario può aggiungere sotto-task$_g$ al task$_g$ assegnatogli.

Ogni sub-task$_g$ deve essere dotato di titolo e descrizione esplicativa. Per partizionare ulteriormente un sub-task$_g$, inserire ogni partizione sotto forma di item di checklist, in cui ogni item corrisponde ad una checkbox

`- [ ] descrizioneItem`.

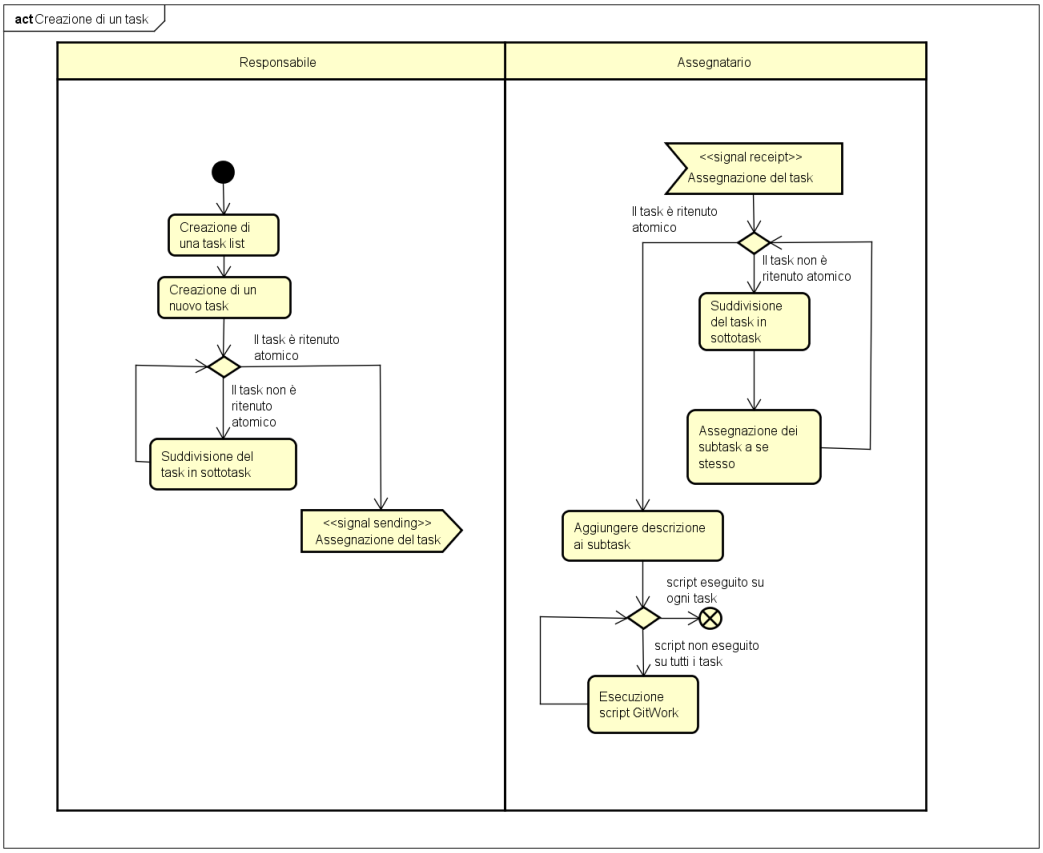


**Figura 9:** Procedura di creazione e gestione di un ticket - Parte 1

Una volta creato il ticket$_g$ su Teamwork$_g$, i passi da seguire sono i seguenti

1. per ogni task$_g$ aperto nella lista precedente, l'assegnatario del task$_g$ esegue l'applicativo *GitWork.jar* che crea, sul repository$_g$ del gruppo *Leaf* una issue$_g$ per ogni subtask del task$_g$ selezionato con la label ToDo;

2. quando il membro del gruppo inizia a lavorare sulla issue$_g$, modifica la label da ToDo a Working;

3. ogni commit eseguito deve riferirsi ad una delle issue$_g$ presenti sul repository$_g$ mediante `#numeroIssue` all'interno del messaggio di commit. Inoltre, è necessario spuntare la relativa checkbox, se presente, per indicare l'avanzamento del proprio lavoro;

4. completato il lavoro richiesto dalla issue$_g$, inserire nel messaggio di commit la stringa `£toVerify`, in modo che la label della issue$_g$ venga modificata da ToDo a ToVerify.

A questo punto, è possibile verificare la issue$_g$ e, di conseguenza, il ticket$_g$.

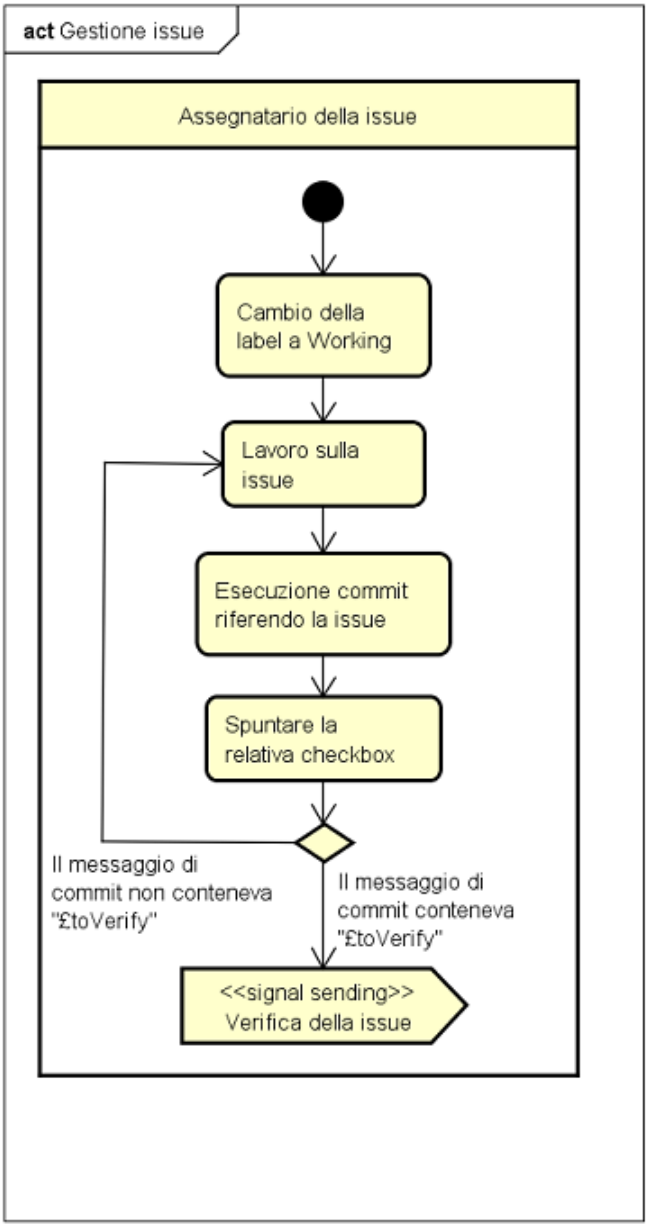


**Figura 10:** Procedura di creazione e gestione di un ticket - Parte 2

**4.1.5.2 Stesura del consuntivo** L'operazione di stesura del consuntivo può essere effettuata solamente dal *Responsabile di progetto* della fase$_g$ corrente.
Le operazioni che il *Responsabile di progetto* esegue sono le seguenti:

1. esporta da Teamwork$_g$ un foglio di calcolo, in formato Microsoft Excel, che mostra le ore rendicontate nella fase$_g$ corrente;
2. recupera il foglio di calcolo (anch'esso in formato Microsoft Excel) relativo alla fase$_g$ corrente, precompilato con il relativo preventivo delle ore e già presente su GitHub$_g$;
3. inserisce le ore rendicontate nelle celle previste all'interno del foglio di calcolo contenente il preventivo delle ore della fase$_g$ corrente, con una semplice operazione di copia-incolla: in questo modo, ottiene la differenza di ore tra il preventivo e le ore rendicontate;
4. aggiunge all'interno della sezione 8 "Consuntivo di periodo" del *Piano di progetto* una sotto-sezione dove si mostrano i valori ottenuti da questo confronto;
5. crea una tabella all'interno di questa sotto-sezione dove mostra:
   - le ore preventivate, suddivise per ruolo, e la differenza tra queste e quelle rendicontate;
   - il budget preventivato, suddiviso per ruolo, e la differenza tra questo e quello ottenuto dalle ore rendicontate;
   - il totale delle ore preventivate per la fase$_g$ corrente, il totale delle ore rendicontate e la loro differenza;
   - il totale del budget preventivato per la fase$_g$ corrente, il totale del budget ottenuto dalle ore rendicontate e la loro differenza.
6. trae infine delle conclusioni dai risultati ottenuti e scrive una valutazione del lavoro effettuato nella fase$_g$ corrente.

### 4.1.6 Strumenti

**4.1.6.1 Pianificazione** Lo strumento scelto per la gestione delle attività di pianificazione di progetto è Teamwork$_g$. Le caratteristiche rilevanti di questo software$_g$ sono le seguenti:

- creazione di attività e sotto-attività assegnabili ad uno o più membri del progetto;
- possibilità di creare dipendenze tra le attività;
- possibilità di assegnare priorità differenti ad ogni attività;
- creazione di milestones da impostare sul calendario;

- creazione automatica di grafici e report, tra cui il diagramma di Gantt$_g$;
- dashboard$_g$ che permette di aver un riepilogo dello stato di avanzamento del progetto, con segnalazione di eventuali ritardi;
- strumento per la segnalazione dei rischi;
- sistema di gestione delle notifiche per ogni attività svolta;
- versione mobile;

Questo strumento, dopo essere stato valutato insieme ad altri strumenti quali Trello, Freedcamp$_g$ e Zoho, è risultato essere il più completo.

#### 4.1.6.2 Creazione dei diagrammi di Gantt

Lo strumento scelto per la realizzazione dei diagrammi di Gantt$_g$ è GanttProject. Le principali ragioni per cui è stato scelto sono:

- gratuito;
- open source$_g$;
- multipiattaforma;
- compatibile con i diagrammi generati da Teamwork$_g$;
- offre la possibilità di creare i diagrammi PERT$_g$;
- offre la possibilità di creare grafici di WBS$_g$;
- generazione dei diagrammi nei formati PDF$_g$, PNG$_g$, HTML$_g$.

#### 4.1.6.3 Calcolo del consuntivo

Per il calcolo del consuntivo si utilizzano due funzionalità offerte da Teamwork$_g$: la rendicontazione delle ore di lavoro da parte di ogni membro del gruppo e l'esportazione delle ore rendicontate tramite un foglio di calcolo in formato Microsoft Excel.
Il *Responsabile di progetto* è colui che effettua la stesura del consuntivo e segue la procedura definita nella sezione 4.1.5.2 "Stesura del consuntivo" del presente documento.
Gli strumenti da lui utilizzati sono i seguenti:

- Teamwork$_g$;
- Microsoft Excel;

## 4.2 Formazione

### 4.2.1 Formazione dei membri del gruppo

Per poter svolgere il progetto didattico, i singoli membri del gruppo necessitano periodicamente di tempo dedicato alla formazione nei vari ambiti che il progetto copre. Chiaramente, non tutte le ore potranno essere a carico del proponente.

**4.2.1.1 Ore rendicontabili e di investimento** Per distinguere se le ore svolte sono a carico del proponente (**rendicontabili**) o per formazione personale (**di investimento**), si può ricorrere al seguente diagramma delle attività.

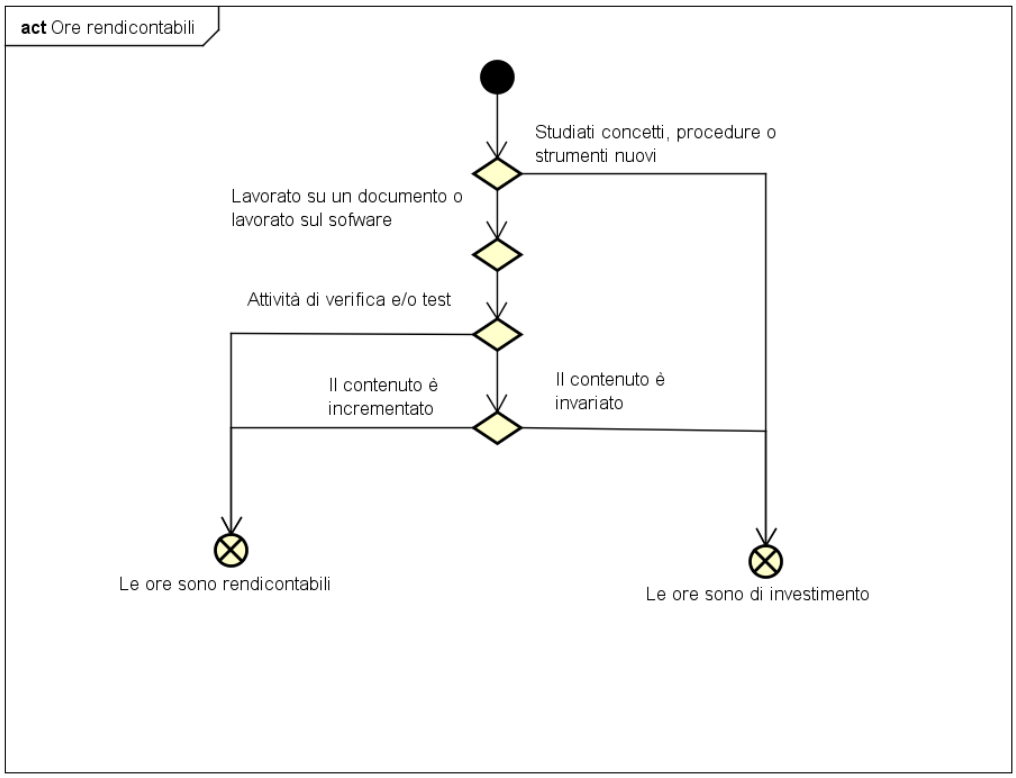


**Figura 11:** Procedura di determinazione delle ore