Norme di progetto

Progetto di Ingegneria del Software A.A. 2025/2026

Versione: v0.4.0

Autore: Atlas
Ultima modifica: 2025/12/12

Tipo di documento: Interno

Registro delle modifiche

Versione	Data	Autore	Verificatore	Descrizione
v0.4.0	2025/12/12	Federico Simonetto	Riccardo Valerio	Fine stesura sez. 2
v0.3.0	2025/12/10	Bilal Sabic	Riccardo Valerio	Inizio stesura sez. 3
v0.2.0	2025/12/09	Federico Simonetto	Riccardo Valerio	Stesura sez. 2
v0.1.0	2025/12/05	Federico Simonetto	Bilal Sabic	Prima stesura introduzione

Indice

1 Introduzione

1.1 Scopo del documento

Lo scopo del documento *Norme di Progetto* è definire le procedure e gli strumenti di lavoro che il team utilizzerà per tutta la durata del progetto didattico.
Per realizzare la stesura del documento, il team ha preso come riferimento lo standard ISO/IEC 12207:1995. Questo divide i processi di ciclo di vita software in tre categorie:

- **Primari**: processi senza i quali un progetto non può dirsi tale;
- **Supporto**: processi che supportano i processi primari;
- **Organizzativi**: processi di carattere più generale, trasversali rispetto ai singoli progetti.

1.2 Glossario

All'interno della documentazione prodotta dal team possono comparire termini suscettibili di incomprensioni o ambiguità. Per evitare questo, è disponibile un glossario contenente i termini tecnici e le loro definizioni. Un termine è consultabile nel glossario se è indicato con la notazione $parola_G$. Premendo sulla G a pedice, l'utente verrà indirizzato alla pagina web del glossario.

1.3 Riferimenti utili

1.3.1 Riferimenti normativi

- Riferimento al capitolato 1 dell'azienda proponente:
  Bluewind S.r.l - Automated EN18031 Compliance Verification
  https://www.math.unipd.it/~tullio/IS-1/2025/Progetto/C1.pdf
- Riferimento al documento dello standard della descrizione dei processi del ciclo di vita software:
  ISO/IEC 12207:1995
  https://www.math.unipd.it/~tullio/IS-1/2009/Approfondimenti/ISO_12207-1995.pdf

1.3.2 Riferimenti informativi

- Riferimento alle slide del corso di Ingegneria del Software:
  Regolamento del progetto didattico
  https://www.math.unipd.it/~tullio/IS-1/2025/Dispense/PD1.pdf
- Riferimento alle slide del corso di Ingegneria del Software:
  Processi di ciclo di vita
  https://www.math.unipd.it/~tullio/IS-1/2025/Dispense/T02.pdf

- Riferimento alle slide del corso di Ingegneria del Software:
  Modelli di sviluppo software
  https://www.math.unipd.it/~tullio/IS-1/2025/Dispense/T03.pdf

- Riferimento alle slide del corso di Ingegneria del Software:
  Gestione di progetto
  https://www.math.unipd.it/~tullio/IS-1/2025/Dispense/T04.pdf

2 Processi Primari

I processi primari comprendono l'insieme di attività che supportano il team di sviluppo e l'azienda fornitrice nell'avvio e nell'esecuzione dello sviluppo, nell'operazione e nella manutenzione di un prodotto software.
Sebbene cinque processi siano classificati come primari, in questo documento approfondiremo i processi di **fornitura** e di **sviluppo**, più interessanti dal punto di vista del progetto didattico.

2.1 Fornitura

Il processo di fornitura descrive le attività e gli incarichi del fornitore che hanno il fine di produrre e consegnare il prodotto software. Il processo ha avvio dopo l'aggiudicazione dell'appalto e prosegue con la definizione delle procedure e delle risorse necessarie per gestire e garantire il progetto, comprendendo lo sviluppo dei piani di progetto e la loro esecuzione, fino alla consegna del prodotto software al committente.

2.1.1 Strumenti utilizzati

Sono stati utilizzati i seguenti strumenti per poter svolgere le attività previste:

- **GitHub:** per il sistema di ticketing, utile per fornire una panoramica generale delle attività previste all'intero team di progetto;
- **Discord:** utilizzato come canale di comunicazione interno;
- **Google Calendar:** utilizzato per tenere traccia delle riunioni, sia interne che esterne;
- **Google Drive:** utilizzato per creare e archiviare i documenti condivisi del gruppo come il foglio delle ore;
- **Google Meet:** utilizzato per partecipare alle riunioni esterne con l'azienda proponente;
- **Telegram:** utilizzato come metodo di comunicazione asincrono con l'azienda proponente.

2.1.2 Attività previste

Il processo di fornitura prevede le seguenti attività:

1. **Avvio:** il fornitore effettua l'analisi delle richieste da parte del proponente, e ne valuta la fattibilità tecnica ed economica tenendo conto delle politiche organizzative e di altre normative applicabili.

2. **Preparazione della risposta:** il fornitore prepara una risposta da inviare alla proponente, includendo le modifiche consigliate emerse durante l'attività di avvio.

3. **Negoziazione:** il fornitore presenta al proponente la risposta formalizzata derivante dall'attività precedente e si avvia una negoziazione finalizzata alla stipula di un contratto.

4. **Pianificazione:** il fornitore conduce un'attività di analisi e consolidamento dei requisiti finalizzata a stabilire un modello di ciclo di vita del software appropriato allo scopo e alla complessità del prodotto. Contestualmente, vengono definiti i requisiti di qualità che il prodotto dovrà soddisfare e viene effettuata un'analisi dei rischi associati a ciascuna opzione di sviluppo.

5. **Esecuzione e controllo:** il fornitore deve attivare e controllare i processi di sviluppo, esercizio e manutenzione, garantendo la qualità e il rispetto dello stato di avanzamento definito dal ciclo di vita stabilito nel contratto.

6. **Revisione e valutazione:** il fornitore organizza le attività di revisione e accettazione del prodotto insieme al proponente, fornendo i report di valutazione, test e revisione previsti dal contratto.

7. **Consegna e completamento:** il fornitore consegna il prodotto software al proponente, assicurando l'assistenza e il supporto previsti dal contratto.

2.1.3 Documentazione fornita

In questa sottosezione vengono elencati i documenti che il team ha sviluppato e si impegna a consegnare ai professori Vardanega e Cardin e all'azienda proponente *Bluewind S.r.l.*.

Analisi dei requisiti	
Redattore	Analisti di progetto
Tipo di documento	Esterno
Scopo e contenuti	Il documento riporta l'elenco dei Casi d'Uso (con relativi diagrammi UML) degli attori coinvolti, i requisiti, funzionali e non, collegati ai Casi d'Uso e al capitolato, e le informazioni relative al loro tracciamento.

Tabella 1: Descrizione del documento "Analisi dei requisiti"

Lettera di presentazione	
Redattore	Responsabile
Tipo di documento	Esterno
Scopo e contenuti	Il documento ha come scopo la presentazione ufficiale ai prof. Tullio e Cardin della candidatura del team alle baseline di progetto.

Tabella 2: Descrizione del documento "Lettera di presentazione"

Norme di progetto	
Redattore	Amministratore
Tipo di documento	Interno
Scopo e contenuti	Documento attuale, riporta gli strumenti e le procedure che il team utilizzerà per tutto lo sviluppo del progetto didattico.

Tabella 3: Descrizione del documento "Norme di progetto"

Piano di progetto	
Redattore	Responsabile
Tipo di documento	Esterno
Scopo e contenuti	Lo scopo del documento è tracciare le attività che vengono svolte così come quelle che vengono pianificate durante tutta la realizzazione del progetto. Il *Piano di progetto* riporta una descrizione di ogni sprint con l'aggiornamento delle ore utilizzate e i relativi costi.

Tabella 4: Descrizione del documento "Piano di progetto"

Piano di qualifica	
Redattore	Amministratore
Tipo di documento	Esterno
Scopo e contenuti	Il documento riporta le strategie di verifica e di validazione che il team di progetto utilizzerà per garantire la qualità del prodotto e dei processi del progetto, oltre ai test effettuati e i rispettivi esiti.

Tabella 5: Descrizione del documento "Piano di qualifica"

2.2 Sviluppo

Il processo di sviluppo si occupa di definire le attività che il team compie per implementare il prodotto software, rispettando le esigenze e le specifiche definite dal proponente.

2.2.1 Strumenti utilizzati

Sono stati utilizzati i seguenti strumenti per poter svolgere le attività previste:

- **Draw.io:** utilizzato per la rappresentazione dei diagrammi degli Use Case;
- **GitHub:** utilizzato come VCS per il prodotto software.

2.2.2 Attività previste

Il processo di sviluppo prevede le seguenti attività:

1. **Implementazione del processo:** lo sviluppatore deve definire e adottare un modello di ciclo di vita del software adeguato agli obiettivi, alla natura e alla complessità del progetto.
2. **Analisi dei requisiti di sistema:** lo sviluppatore deve analizzare il contesto e l'utilizzo previsto del sistema da realizzare, al fine di identificare, definire e documentare in modo completo i requisiti di sistema. Tali requisiti devono includere funzionalità, prestazioni, vincoli operativi e requisiti dell'utente.
3. **Progettazione architetturale del sistema:** lo sviluppatore deve individuare e definire gli elementi hardware e software che compongono il sistema, descrivendone l'architettura a un livello tale da garantire la soddisfazione di tutti i requisiti identificati nella fase precedente.
4. **Analisi dei requisiti software:** per ciascun elemento software lo sviluppatore deve stabilire e documentare i requisiti software, ovvero i requisiti lato utente. Questi comprendono anche le specifiche di qualità.
5. **Progettazione architetturale del software:**lo sviluppatore deve trasformare i requisiti software in un'architettura software che descriva la struttura ad alto livello, i componenti e le loro interazioni.
6. **Progettazione in dettaglio del software:** per ogni elemento software, lo sviluppatore deve produrre una progettazione dettagliata, fino a definire le singole unità software.
7. **Codifica e test del software:** lo sviluppatore deve realizzare e testare ogni unità software, verificando che soddisfi i requisiti assegnati.
8. **Integrazione del software:** lo sviluppatore deve predisporre ed eseguire un piano di integrazione per combinare componenti e unità software in un unico elemento software coerente.
9. **Test di qualifica software:** lo sviluppatore deve condurre test di qualifica per ciascun elemento software, assicurando la conformità ai requisiti di qualifica stabiliti.
10. **Integrazione di sistema:** lo sviluppatore deve integrare gli elementi hardware e software in un sistema completo e operativo.

11. **Test di qualifica del sistema:** lo sviluppatore deve condurre test di qualifica dell'intero sistema, verificando la conformità ai requisiti di sistema.

12. **Installazione del software:** lo sviluppatore deve installare il prodotto software nell'ambiente operativo previsto dal contratto o dalla specifica di progetto.

13. **Supporto all'accettazione software:** lo sviluppatore deve supportare l'utente finale durante le attività di verifica, revisione e test previste per l'accettazione del prodotto.

Il team ha deciso di approfondire di seguito le sole attività di analisi dei requisiti di sistema e codifica del software in vista della **Requirements and Technology Baseline**. Crediamo infatti che, per questa baseline di progetto, le due attività sopra menzionate siano quelle che richiedono una spiegazione più approfondita. Le altre principali attività che il team presenterà verranno aggiunte dopo il "semaforo verde" della RTB.

2.2.3 Attività di analisi dei requisiti

L'analisi dei requisiti è un'attività cardine di ogni progetto e, in particolare, della baseline di progetto Requirements and Technology Baseline. Questa consiste nell'individuare tutti i requisiti, funzionali e non, che il prodotto software deve soddisfare. L'analisi effettuata dal team può essere visualizzata nell'apposito documento.
In particolare, vengono riportati i casi d'uso ed i relativi requisiti secondo la seguente convenzione.

2.2.3.1 Casi d'uso

I casi d'uso sono identificati secondo il seguente format: **UCPrincipale.Alternativo**
dove:

- **UC** è la sigla di Use Case, ovvero Caso d'Uso;
- **Principale** è rappresentato da un numero naturale maggiore di zero e indica che il Caso d'Uso attuale non è correlato ad altri casi d'uso. Il numero indicato dal campo **Principale** è univoco tra tutti i Casi d'Uso;
- **Alternativo** è rappresentato da un numero naturale maggiore di zero e indica che il Caso d'Uso è correlato al Caso d'Uso indicato dal campo **Principale**.

Ogni Caso d'Uso è accompagnato dal relativo diagramma UML, da una descrizione testuale e da un nome che ne riassume lo scenario.

2.2.3.2 Requisiti

I requisiti sono identificati secondo il seguente formato: **RIdTipologiaPriorità**
dove:

- **R** sta per requisito;
- **Id** è un numero naturale maggiore di zero, identificativo univoco di ogni requisito;

- **Tipologia** indica la tipologia del requisito. È rappresentata da uno di questi tre valori:
  - **F**: funzionale;
  - **Q**: qualità;
  - **V**: vincolo.
- **Priorità** indica la priorità del requisito. È rappresentata da uno di questi tre valori:
  - **Ob**: obbligatorio;
  - **D**: desiderabile;
  - **Op**: opzionale.

Ogni requisito viene poi associato al Caso d'Uso cui fa riferimento.

2.2.4 Codifica del software

L'attività di codifica del software viene svolta dai programmatori e rappresenta l'implementazione tramite codice del lavoro effettuato dai progettisti. Il software risultante dall'attività di codifica deve rispettare i requisiti individuati dagli analisti e soddisfare le metriche di verifica e validazione, e deve inoltre soddisfare le esigenze concordate con il proponente.
La codifica deve essere effettuata seguendo specifiche pratiche e diversi standard. Di seguito sono elencate le caratteristiche che il codice sviluppato dal team dovrà rispettare.

2.2.4.1 Stile della codifica

Per garantire uniformità, leggibilità e manutenibilità del codice, il team rispetterà le seguenti regole sintattiche e stilistiche:

- **Convenzioni di Nomeclatura**:
  - Backend:
    - **Classi**: Pascal Case;
    - **Variabili e Metodi**: si adotta lo Snake Case;
    - **Nomi file**: Snake Case;
    - **Costanti**: devono essere scritte interamente in MAIUSCOLO con underscore.
  - Frontend:
    - **Classi**: Pascal Case;
    - **Variabili e Metodi**: si adotta il Camel Case;
    - **Nomi file**: Kebab Case;
    - **Costanti**: uguale al Backend.
- **Formattazione**:
  - **Indentazione**: si utilizzano 4 spazi per ogni livello di annidamento;

  - **Lunghezza righe**: per favorire la lettura su diversi schermi, ogni riga di codice non deve superare i 100 caratteri;
  - **Blocchi di codice**: le funzioni e i blocchi logici devono essere separati da una riga vuota per migliorarne la visibilità.

- **Best practices**:
  - **Lunghezza metodi**: ogni metodo non dovrebbe superare le 20-30 righe di codice; se eccede, va rifattorizzato in sotto-metodi.
  - **Annidamento**: è caldamente consigliato limitare i cicli e le condizioni annidate a un massimo di 3 livelli.
  - **Variabili globali**: l'uso di variabili globali è da evitare dove possibile per prevenire effetti collaterali indesiderati.
  - **Gestione errori**: le anomalie devono essere gestite in modo esplicito tramite l'uso di eccezioni, mantenendo il flusso di lavoro stabile.

- **Lingua e Commenti**:
  - **Lingua**: tutta la codifica, inclusi nomi di variabili, metodi e commenti, deve essere rigorosamente in lingua inglese.
  - **Commenti**: non devono descrivere il "come" (spiegato dal codice stesso), ma il "perché" di una determinata scelta implementativa.
  - **Documentazione (Docstrings)**: ogni funzione o metodo complesso deve essere preceduto da un commento descrittivo che ne specifichi lo scopo e gli argomenti.

2.2.4.2 Strumenti e Tecnologie

- Tecnologie di sviluppo:
  - **Python**: utilizzato come linguaggio di programmazione ad alto livello orientato agli oggetti per la realizzazione della logica di backend. È stato scelto per la sua versatilità e l'ampia disponibilità di risorse.
  - **React**: framework utilizzato per lo sviluppo dell'interfaccia grafica (frontend) dell'applicazione, permettendo un'interazione fluida e responsive.
  - **FastAPI**: framework web moderno, veloce e ad alte prestazioni per la creazione di API RESTful con Python.

- Infrastruttura:
  - **Docker**: piattaforma per eseguire l'applicazione in container, ovvero ambienti isolati che includono tutto il necessario per funzionare correttamente in qualsiasi ambiente di esecuzione. Garantisce la portabilità e la stabilità del sistema indipendentemente dalla macchina locale dello sviluppatore.

- Strumenti di supporto e versionamento:
  - **Git**: Version Control System (VCS) utilizzato per tracciare l'evoluzione del codice sorgente e gestire gli sviluppi in branch separati.
  - **GitHub**: Servizio di hosting per i repository Git che facilita la collaborazione asincrona tra i membri del team, il sistema di ticketing (Issue) e il monitoraggio delle Project Board.
  - **Visual Studio Code (VS Code)**: IDE o editor di testo consigliato per la sua leggerezza e il supporto a estensioni personalizzabili che migliorano la produttività.
- Automazione:
  - **GitHub Actions**: Utilizzate per implementare la Continuous Integration (CI), automatizzando test e verifiche ad ogni commit o pull request per evitare l'introduzione di codice non funzionante nel branch principale.

3 Processi di supporto

3.1 Documentazione

La documentazione è uno strumento potente usato per definire, descrivere, tracciare e supportare le attività svolte durante il progetto. Nel dettaglio, il processo di documentazione supporta la tracciabilità delle informazioni, assicurandone la registrazione e la conservazione lungo tutte le fasi del ciclo di vita delle attività e dei processi.

3.1.1 Strumenti utilizzati

Per compilare la documentazione il gruppo utilizza tre strumenti utili:

- **LaTeX** : un linguaggio mark-up usato per la redazione dei documenti scientifici e altro. Richiede la compilazione ogni volta che si fanno delle modifiche, a differenza di altri linguaggi mark-up come ad esempio Typst. Il gruppo utilizza esclusivamente LaTeX per scrivere tutta la documentazione.
- **Overleaf** : è una piattaforma online che consente di scrivere, modificare e compilare documenti LaTeX direttamente dal browser, facilitando la collaborazione in tempo reale e la gestione dei progetti senza installare software locali. Il gruppo utilizza questo sito per scrivere i documenti piccoli come i verbali.
- **GitHub** : l'adozione di GitHub consente di semplificare i processi di verifica e approvazione dei documenti, garantendo al contempo il controllo delle modifiche tramite le pull request e permette di pianificare i prossimi passi usando il sistema di Ticketing.

3.1.2 Tipologie di documenti

Il gruppo ha creato dei template per ogni tipo di documento e ogni documento ha una struttura specifica. I tipi di documenti con un template sono:

- **Verbale interno**
- **Verbale esterno**
- **Diaro di bordo**

3.1.3 Verbali

Generalmente i verbali seguono questa struttura:

- **Frontespizio**, composto da:
  - Logo e nome del gruppo
  - Tipologia del documento e le informazioni di riferimento, tra cui versione, data e luogo.
  - La lista dei partecipanti alla riunione
- La tabella delle versioni
- Abstract che ha lo scopo di indicare il periodo della riunione e i principali argomenti trattati
- Ordine del giorno che indica in modo strutturato gli argomenti discussi durante la riunione.
- Discussione, dove vengono documentati i temi affrontati, le osservazioni dei partecipanti e le eventuali divergenze o punti di accordo.
- La sezione Decisioni documenta le decisioni prese dal gruppo, organizzate in tabella per facilitare la consultazione e il monitoraggio delle azioni conseguenti.
- La sezione Attività da svolgere elenca in modo dettagliato i compiti assegnati ai partecipanti, specificando i responsabili.

Ogni decisione ha un suo codice univoco e segue questo formato : "Dx (x numero della decisione)-ANNO/MESE/GIORNO_vi" (se interno, ve se interno)
Anche le attività hanno codici univoci e seguono questo formato : "Ax (x numero dell'attività)-ANNO/MESE/GIORNO_vi"

3.1.4 Diario di bordo

Questi documenti, creati nell'ambito dell'attività coordinata dal Prof. Tullio Vardanega, sono utilizzati per monitorare lo stato di avanzamento di ciascun gruppo del I lotto e hanno carattere puramente informativo. Sono presentazioni semplici che trattano tre argomenti.:

- Risultati raggiunti, qui vengono elencate le attività che sono state fate nel periodo corrente
- Prossime attività, un elenco di attività pianificate dal gruppo
- Difficoltà riscontrate, eventuali difficoltà riscontrate durante lo svolgimento delle attività

3.1.5 La produzione

La creazione di un documento prevede questi passaggi:

- Creazione della issue su github tramite sistemi di ticketing: viene aperta una issue per poter tenere traccia dell'andamento e per assegnare il responsabile di quel documento. Per file grandi come ad esempio il PdP vengono creati dei branch secondari per poter effettuare modifiche senza andare a modificare il branch principale. I documenti piccoli come i verbali vanno fatti direttamente in Overleaf senza dover aprire branch secondari.
- Inizio stesura : La creazione del documento è determinata dall'assegnatario: ogni documento è prodotto dal soggetto responsabile delle relative attività.
- Verifica : dopo la stesura il file va verificato tramite pull request di GitHub oppure se è un verbale va verificato direttamente su Overleaf.

Dopo la verifica, il responsabile approva la pull request, e il file viene integrato nel branch principale. Successivamente, il sistema automatizzato GitHub Actions provvede a inserire il file nella cartella corretta del sito web.

3.1.6 Denominazione dei documenti

I documenti, per la denominazione, seguono questo formato:

- **Verbali interni** - VI-ANNO-MESE-GIORNO
- **Verbali esterni** - VE-ANNO-MESE-GIORNO
- **Diari di bordo** - DB-ANNO-MESE-GIORNO

È stato deciso di adottare il formato **AAAA-MM-GG** per le date, al fine di semplificare la lettura e l'ordinamento cronologico dei documenti. La versione del documento si trova all'interno del file.

3.1.7 Manutenzione

È prevista la manutenzione della documentazione quando occorre apportare cambiamenti o aggiornamenti ai contenuti. Per fare la modifica si seguono i prossimi passi :

1. Creare il nuovo branch di lavoro
2. Effettuare le modifiche in locale
3. Dopo aver modificato il file, modificare il registro delle modifiche aggiungendo una nuova riga e fare il push sul branch di lavoro
4. Aprire una pull request per consentire la verifica della qualità del documento
5. Integrare il documento nel branch principale solo se il responsabile conferma la correttezza delle modifiche