UNIVERSITÀ
DEGLI STUDI
DI BERGAMO

Sistema di Analisi dell'Efficienza S.A.E.

Project Plan

Brugnetti Luca - 1086670
Capelli Lorenzo - 1085449
Fustinoni Luca - 1085973

5 dicembre 2024

Indice

1 Introduzione

1.1 Panoramica

SAE24 è un software desktop sviluppato in Java, progettato per **analizzare l'efficienza aziendale**, con particolare attenzione alla **gestione del personale**. SAE24 nasce dall'esigenza delle aziende di monitorare l'efficacia lavorativa del proprio team e di raccogliere dati che consentano di ottenere una visione complessiva delle attività svolte. Questo permette di prendere **decisioni manageriali** più informate e precise. In particolare, il software si occupa di registrare i tempi di realizzazione delle commesse da parte degli operatori, offrendo così un quadro completo delle attività temporali del personale. Sebbene sia pensato principalmente per aziende di medie e grandi dimensioni, SAE24 può rivelarsi utile anche per piccole imprese o start-up che puntano sulla raccolta e analisi dei dati come valore aggiunto.

1.2 Obiettivi del progetto

Gli obiettivi principali del progetto sono:

1.2.1 Registrare i tempi di completamento delle commesse

Ogni dipendente registra il tempo impiegato per completare una task direttamente sul proprio computer.

1.2.2 Monitorare e registrare eventuali interruzioni lavorative

Quando un dipendente sospende una task, seleziona la motivazione dell'interruzione, che viene registrata nel sistema.

1.2.3 Valutare le performance dei dipendenti

L'ufficio competente può monitorare i tempi di realizzazione delle commesse per valutare le capacità e le performance di ciascun dipendente.

1.2.4 Analizzare l'andamento temporale delle commesse

Questa funzionalità consente di analizzare i tempi di completamento delle commesse, identificando aree di miglioramento.

1.2.5 Confrontare le stime temporali degli esperti con quelle derivanti dai dati inseriti dai dipendenti

Questa funzionalità permette di confrontare i tempi effettivi registrati dai dipendenti con le stime fatte dagli esperti, per verificare la loro accuratezza.

2 Modello di processo

Il progetto adotta il framework **Scrum**, un modello di processo Agile che supporta lo sviluppo iterativo e incrementale. La scelta di Scrum si basa sulla necessità di un approccio flessibile e adattabile, ideale per un team di piccole dimensioni come il nostro, composto da soli tre membri.

Scrum consente di affrontare le attività del progetto in modo strutturato, ma senza sacrificare la possibilità di adattarsi ai cambiamenti. Il lavoro è suddiviso in **Sprint**, ciascuno della durata di due settimane.

In particolare per il nostro team, il ruolo di **Scrum Master** sarà assunto a rotazione, garantendo che il framework sia applicato correttamente e che il team rimanga focalizzato sugli obiettivi dello Sprint. Non è prevista la figura del **Product Owner**, in quanto i requisiti del progetto sono definiti e condivisi all'interno del team.

Qui saranno elencati gli **Sprint** con date e Scrum Master associati:

3 Organizzazione del progetto

Il progetto è interamente gestito dal team di sviluppo, con il supporto di due direttori aziendali coinvolti nella fase iniziale di **elicitazione dei requisiti**. Questo approccio garantirà che il software risponda in modo preciso alle esigenze operative e manageriali di un contesto aziendale reale. Le interviste con i direttori saranno strutturate per raccogliere informazioni sui processi lavorativi, le aspettative e i principali obiettivi aziendali. I risultati di queste interviste saranno documentati e integrati nel **Documento di Specifica dei Requisiti**.

3.1 Ruoli e interazioni con esterni

Durante la fase iniziale del progetto, il team di sviluppo si interfaccia con due direttori aziendali per raccogliere i requisiti e le specifiche. L'interazione avviene principalmente attraverso interviste mirate a:

- Identificare i processi aziendali chiave e le loro criticità.
- Determinare le principali funzionalità richieste dal software.
- Definire obiettivi specifici e criteri di successo del sistema.

Questa fase è fondamentale per garantire che il prodotto sviluppato sia allineato con le reali esigenze dell'azienda e contribuisca a migliorare l'efficienza complessiva. Eventuali ulteriori consultazioni con i direttori che saranno effettuate durante il progetto verranno documentate.

3.2 Struttura del team di progetto

Il team di sviluppo è composto da **tre membri**, ciascuno con competenze complementari. Ogni membro ricoprirà tutti i ruoli per garantire la flessibilità necessaria. La struttura del team è così definita:

- **Project Manager:** Responsabile della pianificazione, monitoraggio e gestione complessiva del progetto.
- **Analista:** Gestisce l'elicitazione dei requisiti e documenta le specifiche funzionali e tecniche.
- **Programmatore:** Implementa le funzionalità del sistema utilizzando Java e Swing, seguendo gli standard di qualità.
- **Tester:** Esegue verifiche funzionali e non funzionali per garantire l'affidabilità del software.

L'organizzazione del lavoro è supportata dall'uso di una **Kanban board**: sarà utilizzata la Kanban board fornita da **GitHub Projects**, che permetterà al team di collaborare in modo efficiente, gestendo il flusso di lavoro, assegnando compiti e monitorando l'avanzamento delle attività. La board sarà strutturata con colonne che rappresentano le diverse fasi del processo di sviluppo ("To do", "In progress", "Done"), facilitando così una gestione visibile e dinamica delle attività.

4 Standard, linee guida, procedure

Il progetto seguirà gli standard di codifica definiti da **Oracle** per Java. La documentazione sarà sviluppata seguendo le convenzioni **UML**, assicurando chiarezza e coerenza nella rappresentazione del sistema. Verranno utilizzati diagrammi UML per modellare requisiti, progettazione e strutture dati, favorendo una piena comprensione del sistema da parte di tutti i membri del team. La specifica dei requisiti (SRS) seguirà lo standard **IEEE830**. Nell'ambito delle linee guida organizzative, è stabilito che l'intero sistema sarà implementato esclusivamente all'interno dell'**IDE Eclipse**, senza ricorrere a strumenti o servizi esterni.

5 Attività di gestione

Il team terrà riunioni settimanali informali sia in presenza sia tramite Discord per discutere rapidamente dell'avanzamento del progetto, risolvere eventuali problematiche urgenti e coordinarsi sulle attività quotidiane. In aggiunta, si terranno **sprint review** al termine di ogni sprint, per fare il punto sui risultati ottenuti e pianificare i passi successivi.

Le richieste di cambiamento saranno raccolte tramite issue su **GitHub Project**, assegnando priorità specifiche. Ogni modifica verrà sviluppata in un branch dedicato e successivamente sottoposta a pull request per una revisione collettiva e l'integrazione nel progetto principale.

6 Rischi

Il progetto presenta i seguenti rischi principali:

6.1 Mancanza di disponibilità di tempo dei membri del team

La limitata disponibilità di tempo di ciascun membro del team, dovuta a impegni accademici e personali, potrebbe portare a ritardi nella realizzazione delle attività pianificate, compromettendo le scadenze.

6.2 Difficoltà nell'utilizzo di nuovi strumenti e tecnologie

L'introduzione e l'apprendimento di strumenti come Maven, Papyrus e Swing potrebbe rallentare le attività di sviluppo, soprattutto in assenza di una familiarità preesistente con queste tecnologie. Errori dovuti alla scarsa conoscenza degli strumenti potrebbero influire negativamente sull'efficienza del lavoro.

6.3 Integrazione e compatibilità delle componenti software

La combinazione di diverse tecnologie (Java, Swing, Maven) e strumenti di gestione (GitHub, Papyrus) potrebbe generare problemi di compatibilità o errori nell'integrazione finale del progetto, con conseguente aumento dei tempi di sviluppo e testing.

6.4 Ritardi nell'approvazione dei cambiamenti

Il processo di gestione delle modifiche tramite GitHub, che richiede la revisione e l'approvazione collettiva, potrebbe causare rallentamenti, specialmente in caso di divergenze o tempi di risposta dilatati.

7 Personale

Oltre al team di sviluppo non sarà presente ulteriore personale.

8 Metodi e tecniche

8.1 Ingegneria dei requisiti

Per l'elicitazione dei requisiti, utilizzeremo tre principali tecniche:

- **Intervista:** Realizzeremo interviste con i direttori aziendali per raccogliere le esigenze specifiche e comprendere i processi aziendali chiave. .
- **Brainstorming:**Attraverso riunioni in presenza discuteremo sui requisiti e le loro priorità.

Inoltre, seguiremo lo standard **IEEE 830** per la stesura del **Documento di Specifica dei Requisiti** (SRS) e utilizzeremo la tecnica **MoSCoW** (Must have, Should have, Could have, Won't have) per la prioritizzazione dei requisiti.

8.2 Modellazione e architettura del software

Per la modellazione dell'architettura del software, utilizzeremo **Papyrus**, uno strumento di modellazione UML che ci permetterà di creare diagrammi dei casi d'uso, delle classi, delle sequenze e di altre strutture necessarie per definire chiaramente l'architettura e le interazioni del sistema.

8.3 Testing

Per garantire la qualità del software, utilizzeremo **JUnit**, un framework per la realizzazione di test unitari, che ci permetterà di eseguire test automatizzati sulle singole unità di codice, garantendo che ciascun componente del sistema funzioni correttamente.

9 Garanzia di qualità

Per garantire la qualità del software sviluppato, seguirà i criteri di qualità definiti dal modello di McCall, che include aspetti fondamentali come la correttezza, l'affidabilità, l'efficienza, la manutenibilità e la portabilità del sistema.

Per l'analisi statica del codice e il miglioramento continuo della qualità, utilizzeremo strumenti come:

- **STAN4j:** Un tool di analisi statica per individuare errori nel codice prima dell'esecuzione.
- **SonarLint:** Un plugin integrato negli IDE per l'analisi in tempo reale del codice, che aiuta a identificare e correggere i problemi di qualità durante lo sviluppo.

Questi strumenti contribuiranno a migliorare la qualità complessiva del progetto, supportando il team nel rispetto dei criteri di McCall.

10 Pacchetti di lavoro (workpackages)

Il progetto è suddiviso nei seguenti pacchetti di lavoro:

- Riunioni e interviste ai direttori aziendali
- Redazione della documentazione
- Creazione dei diagrammi UML
- Sviluppo del codice
- Creazione del database
- Testing

11 Risorse

Durante le fasi del progetto verrano impiegati i seguenti tool:

- **Overleaf** (stesura del project plan in Latex)
- **Discord** (riunioni non in presenza)
- **Eclipse** (programmazione software in Java)
- **Papyrus** (creazione diagrammi)
- **JavaFx** (grafica progetto)
- **Maven** (gestione dipendenze)
- **EclipseLink** (ORM)

12 Budget

Non è previsto un costo di sviluppo del progetto. Il budget temporale è calcolato dalla somma delle ore che ciascun membro del team si impegna a mettere a disposizione del progetto, per un totale di 120 ore.

13 Cambiamenti

Essendo un team che adotta il framework Agile, i cambiamenti fanno parte integrante del processo di sviluppo. L'approccio iterativo e incrementale ci consente di adattarci alle nuove esigenze e di rispondere rapidamente a eventuali modifiche ai requisiti o alle priorità.

Come descritto in precedenza, i cambiamenti verranno gestiti attraverso il sistema di gestione delle issue su GitHub. Ogni modifica proposta sarà registrata come issue, a cui verrà assegnata una priorità specifica e una descrizione dettagliata. Per garantire una gestione strutturata e trasparente, ogni cambiamento sarà implementato in un branch apposito, il cui nome indicherà chiaramente l'issue associata.

14 Consegna

Modalità: Il progetto deve essere consegnato al professor Gargantini Angelo tramite repository GitHub. Insieme al progetto sarà consegnata la relativa documentazione.

Data: La data di consegna è fissata a 5 giorni prima dell'esame quindi il team si impegna a terminare il progetto entro la data 19/01/2025.