Ingegneria del Software

Dipartimento di Ingegneria e Scienza dell'Informazione

Gruppo G-51: Giuliano Campagnolo, Elisa Beltrame, Mattia Marini

Mergify - merge your events

Deliverable 5

Report finale

Anno accademico 2022/2023

Università di Trento

Indice

1 Obiettivi del documento

Questo documento presenta un'analisi approfondita dell'organizzazione del lavoro, compresi i ruoli assegnati, il tempo dedicato da ciascun membro, le criticità riscontrate e un'autovalutazione complessiva.

2 Approcci all'ingegneria del Software

2.1 Seminario Blue Tensor

Jonni Malacarne, CEO di Blue Tensor, ha illustrato i metodi di ingegneria del software applicati allo sviluppo di progetti di intelligenza artificiale (AI). Il team di sviluppo, composto da diverse figure tra cui Project Manager, Senior Software Architect, Ai Engineers e altri, utilizza un approccio basato sui Work Package (WP). Questi WP sono suddivisi in cinque operativi e uno trasversale:

- Analisi e Fattibilità (WP1): Il team collabora con il cliente per definire le funzionalità richieste e identificare le criticità del progetto, stabilendo requisiti funzionali e non funzionali.
- Setup e Design (WP2): Si progetta l'architettura del sistema, includendo l'infrastruttura hardware e software. Si utilizzano tecniche come lo Story Mapping per definire le funzionalità desiderate.
- Data Preparation (WP3): Si selezionano e preparano i dati necessari per l'addestramento dei modelli di intelligenza artificiale, creando dataset di training e di test.
- Sviluppo - Training - Test - Deploy (WP4): Il team sviluppa il software in modalità Agile, rilasciando piccole porzioni di software testabili. Si garantisce un feedback tempestivo per soddisfare le esigenze del cliente.
- Final Release (WP5): Si procede al rilascio finale del progetto, includendo la raccolta dei feedback e l'implementazione di eventuali miglioramenti identificati durante il processo.

Questo approccio strutturato e metodico consente al team di gestire il progetto in modo efficiente, garantendo il raggiungimento degli obiettivi e la soddisfazione del cliente.

2.2 Simulazione Metodo Kanban

Nel seminario tenuto dal Dr. York Rössel, si è andato a simulare un metodo di ingegneria del Software ideato da David J. Anderson: il metodo Kanban, che consiste in un approccio di gestione del lavoro basato sulla visualizzazione visiva dei processi per migliorare trasparenza, comunicazione ed efficacia. Le sue caratteristiche principali includono la creazione di una board divisa in colonne rappresentanti le varie fasi del progetto, come Design, Implementation e Done. Ogni parte del lavoro è simboleggiata da un post-it che, una volta completato, viene spostato alla colonna corrispondente, fino al completamento. Inoltre, per ogni colonna di lavoro, si stabilisce un limite massimo di post-it consentiti per mantenere un flusso costante e identificare inefficienze nel processo. Il metodo Kanban facilita il rilevamento tempestivo di ritardi e blocchi nel progetto, consentendo una visione complessiva del lavoro e un adattamento continuo alle esigenze e ai cambiamenti nel tempo. Questa flessibilità favorisce ulteriori miglioramenti e adattamenti, rendendo il processo più efficiente e adattabile.

2.3 Seminario IBM

Alessandro Gorga, rappresentante tecnico aziendale di IBM, illustra il potenziale del cloud come un vasto negozio di tecnologie software distribuite, fornendo potenza computazionale e servizi software accessibili a tutti. Il Cloud IBM affronta le sfide attuali fornendo un contesto sicuro per eseguire sistemi software, garantendo segretezza attraverso la crittografia avanzata e offrendo tecnologie open source all'avanguardia. Inoltre, favorisce l'utilizzo sostenibile dell'informatica industriale ottimizzando i consumi e riciclando il calore. Il Cloud IBM offre un ambiente ideale per lo sviluppo collaborativo del software, consentendo ai team di lavorare insieme in modo efficiente e flessibile. Attraverso una toolchain integrata nel catalogo del Cloud IBM, è possibile creare un ambiente di sviluppo completo che comprende strumenti per la gestione del codice, la compilazione, i test e il rilascio del software. La toolchain supporta l'integrazione continua (CI) e la distribuzione continua (CD), permettendo al team di eseguire automaticamente test e rilasci ogni volta che viene apportata una modifica al codice sorgente. Questo processo automatizzato garantisce la rapidità e l'affidabilità delle modifiche apportate al software, riducendo al minimo il rischio di errori e semplificando il processo di sviluppo. Inoltre, il Cloud IBM offre una vasta gamma di servizi e risorse che possono essere integrati nella toolchain per migliorare le capacità di sviluppo del software. Ad esempio, è possibile utilizzare servizi di analisi dei dati per ottenere insight utili dal proprio software, o servizi di intelligenza artificiale per aggiungere funzionalità avanzate.

2.4 Seminario META

Heorghi Raik condivide le metodologie di Software Engineering all'interno di Meta, un'azienda multinazionale. I Software Engineering in Meta sono responsabili della realizzazione dei progetti e hanno diverse responsabilità. All'inizio della loro carriera in Meta, si concentrano principalmente sulla programmazione, ma con l'esperienza acquisita, possono avanzare verso ruoli di Project Manager, una pratica non comune in tutte le aziende ma adottata in Meta. Le loro responsabilità tecniche includono la programmazione, la progettazione architettonica, il project management e altro ancora. Oltre alle responsabilità tecniche, i Software Engineering hanno anche compiti non tecnici, come la gestione delle risorse umane per il progetto, la promozione del team building e l'assunzione del ruolo di mentore o coach per altri membri del team. Rispetto alla metodologia di lavoro, Meta non impone un metodo specifico, lasciando a ciascun gruppo di lavoro la libertà di decidere come organizzare il proprio lavoro. Inoltre, non esiste un linguaggio di programmazione ufficiale all'interno dell'azienda; i dipendenti possono utilizzare il linguaggio che preferiscono. Tuttavia, è consigliabile utilizzare linguaggi noti per garantire la comprensibilità del codice. Tra i linguaggi più utilizzati vi sono Hack, JavaScript, Python e SQL. Questa flessibilità consente ai dipendenti di adattare il proprio lavoro alle esigenze specifiche del progetto e alle proprie competenze linguistiche.

2.5 Seminario U-Hopper

Daniele Miorandi, CEO di U-Hopper, guida un'azienda specializzata nell'analisi di big data e Intelligenza Artificiale. Utilizzano tecnologie open source e infrastrutture cloud per gestire grandi quantità di dati. I linguaggi principali sono Python, Scala e Go, con focus sui microservizi e sistemi asincroni per l'efficienza e la scalabilità. Il processo di sviluppo di U-Hopper prevede la suddivisione del codice in sprint iterativi, definendo milestone, issue e scadenze. Il processo di sviluppo inizia definendo una milestone e suddividendo il lavoro in una serie di issue, stabilendo scadenze per ogni fase. Questo framework guida lo sviluppo degli sprint, partendo dalla fase di codifica, seguita dalla definizione dei test (con una copertura minima del 70%) e la revisione del codice tramite Merge Requests per garantirne la solidità. I deployment avvengono attraverso diversi ambienti di test, con un Deployment staging che simula l'ambiente di produzione. Una volta superati i test, il codice viene distribuito in produzione, con una particolare attenzione al monitoraggio continuo e alla gestione delle anomalie attraverso il rollback automatico. Il monitoraggio e il controllo dell'operato sono fondamentali in U-Hopper, in quanto garantiscono il funzionamento ottimale del software e la rilevazione tempestiva di eventuali problemi. La metodologia adottata consente una gestione efficace e flessibile dei progetti, assicurando un alto standard di qualità e performance.

2.6 Seminario RedHat

Durante il seminario condotto da Mario Fusco, Drools Project Lead, sono stati esplorati i vantaggi dei progetti Open Source. Tra questi, spiccano la condivisione della conoscenza, la revisione da parte dei pari, il processo meritocratico e la visibilità offerta dalla disponibilità del codice a tutti. Un Software Open Source richiede la tutela delle licenze, che possono essere di tipo Copyleft o non-Copyleft. I clienti scelgono il Software Open Source per il costo più basso e la maggiore flessibilità. Partecipare a un progetto Open Source implica avviare discussioni, dimostrare il valore delle proprie contribuzioni e raggiungere un accordo con il responsabile per la fusione delle modifiche. Nei progetti Open Source, si cerca attivamente di coinvolgere nuove persone, spesso segnalando aree in cui è necessario aiuto tramite etichette come "Help Wanted". Questo può generare una maggiore pressione per partecipare, specialmente quando si tratta di compiti che possono rallentare altri progetti o processi di lavoro. Al contrario, quando un compito è classificato come Low Priority, c'è meno urgenza e più tempo per contribuire. La partecipazione ai progetti Open Source avviene su vari livelli, ognuno con compiti specifici: Novice (segnalazione di bug), Beginner (correzione e traduzione della documentazione), Enchanter (assistenza ai principianti per aumentare la loro velocità), Druid (risoluzione di bug), Magician (contributo a nuove funzionalità e revisione del lavoro degli altri) e Wizard (membro chiave del team).

2.7 Seminario Microsoft

Diego Colombo, della Development Division di Microsoft, è il presentatore di questo seminario che esplora il mondo del testing software. Il testing consiste nel verificare la correttezza di un pezzo di codice attraverso una combinazione di verifica manuale e automatizzata, sia per i requisiti funzionali che non funzionali. Un bug è un comportamento del software che sorprende o frustra l'utente, rappresentando un errore nel sistema. Il testing avviene a diversi livelli e ha diverse connotazioni. L'esempio più conosciuto sono gli Unit Test, noti per la loro natura booleana, ma altri tipi di test possono fornire risultati più variabili e non necessariamente binari. Il Benchmarking test, infatti, si distingue per l'assenza di questo binarismo. Quando superiamo questo tipo di test, il risultato diventa il punto di riferimento per valutare le prestazioni relative e individuare potenziali miglioramenti. Acceptance Test, invece, è strettamente legato all'accettazione del software in termini di requisiti aziendali e comportamento del sistema. L'importanza di testare il software è cruciale: il processo produce risultati, rappresentati da un report, che indicano quanto del codice è coperto da questi test, concetto noto come Code Coverage. Ma i test non sono solo per prevenire bug o evitare regressioni; sono anche uno strumento per progettare, esplorare, modellare e documentare. I test devono essere eseguiti in ogni fase del processo, dall'iniziale fase di test sulla propria macchina fino alla fase automatizza-

ta. È fondamentale mantenere coerenza e stabilità nel processo di testing, assicurando uniformità tra i test e il codice che vanno a verificare, indipendentemente dal livello di operatività.

2.8 Seminario Molinari

Andrea Molinari ha tenuto un seminario focalizzato sui sistemi legacy, con particolare attenzione ai mainframe, ampiamente utilizzati nel settore bancario, assicurativo e finanziario. Durante l'analisi dei sistemi legacy, sono state esaminate le principali tipologie, evidenziando le loro caratteristiche, i vantaggi e le sfide rispetto agli ambienti più moderni, nonché le implicazioni sul debito tecnologico. Il debito tecnologico, o tech debt, si riferisce alla conseguenza di scelte tecnologiche che comportano costi aggiuntivi nel lungo termine, come ad esempio il mantenimento di sistemi obsoleti. Nei sistemi legacy, è comune trovare sistemi operativi obsoleti integrati con hardware specifici. Per quanto riguarda i linguaggi di programmazione, Cobol mantiene una forte presenza, accanto a Pascal, Java e altri. Attualmente, molte aziende offrono soluzioni per modernizzare le applicazioni e migrare nel cloud, ma ci sono numerosi ostacoli che frenano questo cambiamento, come i costi di migrazione, la mancanza di competenze specifiche e la resistenza del personale non adeguatamente formato al cambiamento. Le attuali strategie per affrontare il problema includono l'integrazione dei mainframe in ambienti ibridi o misti con il cloud, nonché la conversione o riscrittura del codice legacy. Tuttavia, il mondo legacy non è sempre compatibile con metodologie di sviluppo agili e spesso si preferiscono approcci più predittivi. Il seminario ha offerto una panoramica sulle tecnologie attualmente in uso in molte realtà, sottolineando la sfida significativa rappresentata dal tentativo di modernizzare le applicazioni legacy in modo efficiente e ottimale, integrando le nuove tecnologie.

2.9 Seminario Marsiglia

Edoardo Marsiglia, Enterprise architect, conduce un seminario sull'ingegneria del software, concentrandosi sulla modernizzazione delle applicazioni. Questo processo mira a migliorare l'efficienza, la scalabilità, la sicurezza e la conformità agli standard attuali delle applicazioni esistenti. All'interno delle aziende, diversi ruoli contribuiscono alla gestione e all'evoluzione delle applicazioni, come lo specialista, l'architetto, il Consultant, l'ingegnere e il Manager. Le architetture delle applicazioni possono essere rappresentate con una struttura piramidale, con il tecnologico alla base, seguito dall'applicazione, dai dati e infine dal business. La modernizzazione delle applicazioni rappresenta un importante passo verso il miglioramento delle operazioni aziendali e l'adattamento alle esigenze del mercato in evoluzione. Le aziende esplorano nuove soluzioni e approcci per ottimizzare

le loro applicazioni esistenti, sfruttando al massimo le architetture software moderne, le nuove tecnologie e le potenzialità offerte dal cloud. Questo processo non solo migliora le prestazioni e la sicurezza delle applicazioni, ma apre anche nuove opportunità di innovazione e crescita per le imprese.

2.10 APSS & Trentino.ai

Il Dipartimento Tecnologie costituisce un pilastro fondamentale nell'ambito dell'azienda sanitaria, focalizzandosi su vari settori, tra cui infrastrutture, applicazioni, data management e ingegneria clinica. Il suo obiettivo primario è massimizzare i benefici delle tecnologie impiegate, puntando verso la convergenza verso il cloud. Questo approccio strategico mira a migliorare l'efficienza complessiva, riducendo i rischi e garantendo la sicurezza dei dati, elementi essenziali nel contesto sanitario. Durante il seminario dedicato all'Azienda Sanitaria, sono stati presentati i principi fondamentali che guidano l'ingegneria del software nell'ambito sanitario. Tra questi principi, spicca l'adozione di un approccio "waterfall" anziché AGILE. Questa scelta è motivata dalla necessità di gestire il budget in modo efficace e garantire la continuità dei processi senza interruzioni, aspetti critici nel contesto sanitario. Parallelamente, l'azienda si impegna nella gestione dei dati attraverso attività di filtraggio e analisi. Un obiettivo prioritario è l'implementazione di un fascicolo sanitario elettronico (FSE) basato sui dati, il quale richiede una ristrutturazione delle informazioni attualmente presenti in documenti tradizionali. In conclusione, l'Azienda Sanitaria adotta un approccio proattivo, orientato alla stabilità e alla sicurezza nei processi. Tuttavia, questo comporta inevitabilmente una certa rigidità e una limitata flessibilità, elementi che vengono valutati attentamente nel contesto della modernizzazione e ottimizzazione dei servizi sanitari.

2.11 Gestione del progetto Mergify

Dinanzi alle difficoltà riscontrate dai vari membri del gruppo nell'incontrarsi frequentemente in persona, abbiamo ritenuto opportuno suddividere il carico di lavoro in pacchetti e task. Al fine di mantenere una visione d'insieme dello sviluppo dell'applicazione web, abbiamo adottato il metodo Kanban. Tuttavia, anziché utilizzare una "board" fisica, abbiamo preferito avvalerci di un documento Excel, diviso in quattro fasi: "inizio", "elaborazione", "revisione", "fine". Seguendo la logica del metodo, abbiamo assegnato a ciascun membro del gruppo diversi pacchetti. Ogni qualvolta un membro iniziava un nuovo pacchetto, lo annotava sull'Excel, corredandolo con una breve descrizione del lavoro in corso. Qualsiasi modifica apportata veniva tempestivamente condivisa con gli altri membri del gruppo. Al completamento di un certo numero di pacchetti, ci incontravamo per valutare i progressi e i risultati ottenuti. Tale approccio ci ha permesso di mantenere

tutti i membri del gruppo costantemente aggiornati sullo stato del progetto, riducendo il rischio di incomprensioni e consentendo di individuare tempestivamente eventuali problematiche.

3 Organizzazione del Lavoro

Il progetto è stato condotto con impegno da parte di tutti i membri del gruppo, i quali hanno lavorato principalmente in modo autonomo. Ciononostante il team si è incontrato ogni due settimane per fare il punto sulle attività svolte, valutare i progressi fatti e assegnare nuovi compiti. Come descritto in Tabella 1, abbiamo scelto di suddividere il lavoro in base alle competenze specifiche e agli interessi di ciascun membro. La ragione per questa scelta è quella di rendere piú efficiente lo svolgimento dei lavori e sfruttare le competenze specifiche di ciascun membro. Ciò ha garantito un'equilibrata partecipazione di ogni membro ed ha evitato il sovraffaticamento e il conseguente deterioramento delle prestazioni. Gli incontri si sono tenuti sia online (tramite riunioni Telegram) che in presenza a Povo, specialmente subito dopo le lezioni, con lo scopo di individuare eventuali problemi e trovare soluzioni appropriate. Abbiamo utilizzato Microsoft Excel per tenere traccia del contributo dei membri del gruppo e organizzare il lavoro efficacemente.

4 Ruoli e Attività

Componente del team	**Ruolo**	**Principali attività**
Giuliano Campagnolo	Progettista tecnico, relatore e tester	Si è occupato principalmente alla stesura dei diagrammi e al testing. Ha contribuito alla stesura e alla revisione dei Deliverables, in particolare D3 e D4.
Mattia Marini	Designer, sviluppatore front-end, back-end e documentazione	Designer del gruppo, principale progettista del prototipo e successivamente programmatore di front, back-end e strumenti di utility, documentazione e testing. Ha contribuito nella stesura dei deliverables e nella loro impaginazione.
Elisa Beltrame	Product Manager	Si è occupata della scrittura dei documenti e della loro revisione, in particolare del D1, D2 e D5. Inoltre ha collaborato nella progettazione del prototipo.

Tabella 1: Ruoli e principali attività del team

5 Carico e Distribuzione del Lavoro

L'analisi dei file di log ha evidenziato la distribuzione del carico di lavoro, espresso in ore/persona, per ciascun membro del gruppo. Tale distribuzione è stata progettata con l'obiettivo di assicurare una suddivisione equa del lavoro, basata principalmente sulle competenze individuali di ciascun membro. Un riepilogo delle ore è riportato nella Tabella 2.

Componente del gruppo	**D1**	**D2**	**D3**	**D4**	**D5**	**TOT**
Giuliano Campagnolo	15h	10h	6h	84h	6h	121h
Mattia Marini	24h	15h	4h	131h	3h	177h
Elisa Beltrame	23h	18h	6h	56h	20h	123h
TOT	52h	37h	16h	234h	32h	371h

Tabella 2: Ore di lavoro per componente del gruppo

6 Criticità

All'inizio del progetto, abbiamo subito affrontato una sfida: l'organizzazione degli incontri per lavorare collettivamente. Ciò ha causato ritardi nel progresso complessivo, in particolare nella stesura del D1 come riportato nella Tabella 2. In seguito, abbiamo adottato un approccio organizzativo diverso, preferendo suddividere il lavoro in piccole parti e lavorare autonomamente per il raggiungimento degli obiettivi del progetto. Questo cambiamento ha portato a una significativa riduzione delle ore impiegate per il D2 e il D3, accelerando così il ritmo di lavoro e migliorando la qualità del prodotto finale.
Per quanto riguarda il D4, ci sono state difficoltà nella scrittura del codice, vista l'assenza di conoscense pregresse da parte dello sviluppatore, le quali hanno portato ad un ritardo secondo quanto previsto, complice anche l'inizio della sessione invernale. Per evitare ulteriori ritardi, abbiamo quindi deciso di riprendere prima i seminari del D5.

7 Autovalutazione

Durante l'intero progetto, abbiamo mantenuto un costante impegno lavorativo. Nonostante i problemi riscontrati e il differente numero di ore impiegato, ogni membro del gruppo ha contribuito a sostenere gli altri. L'organizzazione, la suddivisione e il metodo di lavoro scelto hanno permesso al nostro gruppo di superare queste difficoltà con successo, provando l'indiscutibile valore aggiunto di un lavoro in team in una simulazione di contesto lavorativo. Sulla base delle considerazioni espresse, la valutazione finale che noi riteniamo opportuna è riportata nella Tabella 3.

Componente del gruppo	**Voto**
Giuliano Campagnolo	30
Mattia Marini	30
Elisa Beltrame	30

Tabella 3: Valutazione finale del gruppo