UNIVERSITÀ DI TRENTO

Dipartimento di Ingegneria e Scienza dell'Informazione

Progetto:

Co.Inqui

Titolo del documento:

Report Finale

Document Info:

Doc. Name	D5_Co.Inqui_ReportFinale		
Doc. Number	D5	Group number	G30
Description	Report Finale del progetto: approccio all'Ingegneria del software, organizzazione del lavoro, ruoli, tempo complessivo e di ciascun membro dedicato al progetto, criticità, autovalutazione		

SCOPO DEL DOCUMENTO

In questo ultimo documento analizzeremo inizialmente diversi approcci all'ingegneria del software sperimentati da persone facenti parte di diverse realtà e aziende, i quali ci hanno appunto presentato le loro personali esperienze mediante dei seminari e successivamente andremo ad esporre l'approccio che è stato utilizzato per la realizzazione di questo progetto, andando, in secondo luogo, a specificare nel dettaglio l'organizzazione del lavoro, la suddivisione dei ruoli all'interno del team, il carico di lavoro di ciascun membro e le criticità incontrate. Concluderemo, infine, con una breve autovalutazione.

APPROCCI ALL'INGEGNERIA DEL SOFTWARE

In questo paragrafo andremo a sintetizzare brevemente il contenuto di ciascun seminario, mettendo in particolare risalto i diversi approcci all'ingegneria del software. Concluderemo, infine, con una descrizione del nostro approccio illustrando similarità e differenze riscontrate rispetto alle esperienze che ci sono state presentate nel corso del semestre.

BLUE TENSOR

Blue Tensor è un'azienda trentina che si occupa di realizzare soluzioni e prodotti personalizzati basati sull'intelligenza artificiale. Queste soluzioni sono fortemente pensate per l'industria, con lo scopo di supportare decisioni, ottimizzare la produzione e gestire il know-how aziendale.
Ci sono state presentate le forti analogie tra il metodo di sviluppo di un progetto AI e quello per sviluppare un software tradizionale: si parte dall'analisi dei requisiti e dal capire cosa vuole il cliente, poi si passa all'analisi di fattibilità e un ipotetico Proof of Content, ovvero una sorta di prototipo da mostrare al cliente. La parte più consistente del lavoro si basa, poi, sullo sviluppo a livello pratico, e quindi di codice, del prodotto, ovvero la fase di "Engineering" e infine si ha una fase di miglioramento continuo dal momento in cui la soluzione viene effettivamente messa all'opera. Da quel momento, infatti, essa stessa raccoglierà i dati necessari a ri-educare gli algoritmi scritti originariamente.
Nel caso dell'intelligenza artificiale bisogna fare ulteriormente attenzione a determinati aspetti, come la quantità e la qualità dei dati, inoltre è necessario avere un team eterogeneo e con diversi background affinché queste figure possano aiutare gli sviluppatori a capire le necessità del cliente e i tecnicismi dell'ambito in cui verrà collocato il prodotto.
Per le prime fasi del progetto Blue Tensor tende ad affidarsi ad una metodologia a cascata per poter fornire un'analisi dei costi alle industrie, solo successivamente si passa alla metodologia agile che diventa nettamente più pratica nella fase di sviluppo.
Abbiamo notato anche una forte somiglianza per quanto riguarda gli strumenti utilizzati, quali diagrammi UML, diagrammi ER per i database, l'uso di mockup per visualizzare la User Interface.

KANBAN SIMULATION

La simulazione del metodo Kanban è stata un'esperienza interattiva che aveva lo scopo di farci entrare in una diversa visione della gestione del lavoro.
Kanban ci è stato presentato come un metodo che, tramite principi, pratiche e tabelle, permette di visualizzare in maniera più concreta la mole di lavoro da eseguire e il flusso, quindi l'ordine, in cui vogliamo che le singole parti vengano

svolte, in modo da guidare l'organizzazione e suddivisione delle mansioni al fine di ottimizzare i tempi.
Durante la simulazione ci sono parsi subito parecchio rilevanti i meeting giornalieri per decidere con i compagni di gruppo quali mosse fosse meglio attuare per ottimizzare il workflow, e i "WIP limits", ovvero i limiti di Work In Progress, che hanno favorito la coesione del team, la cooperazione tra membri e l'aumento di concentrazione su determinati compiti con la conseguente diminuzione dei tempi di svolgimento.
Anche in questo seminario è risultata fondamentale la capacità di adattamento, tipica della modalità di lavoro "agile".

IBM

IBM è l'azienda informatica più antica al mondo, fondata nel 1911, conosciuta per la sua innovazione, affidabilità e serietà.
Il seminario è stato incentrato sulla spiegazione dei vari servizi che offre il Cloud IBM, una piattaforma che fornisce sia potenza computazionale, sia tecnologie software a pagamento.
Ci è stato presentato quindi questo utile mezzo, strategico per il rinnovamento dei sistemi software, in quanto fornisce diversi strumenti e API utili per gestire le nostre webapp e applicativi software. Inoltre, navigando nel loro sito web abbiamo trovato anche diversi elementi DevOps potenzialmente utili per la gestione del workflow o per tenere traccia del ciclo di produzione e vita di un prodotto.
Anche in questo caso, mediante l'introduzione dei DevOps si è parlato di approccio "Agile" all'ingegneria del software.

META

Il seminario di Meta è stato condotto da un ex studente dell'università di Trento.
Alcuni aspetti che hanno attirato la nostra attenzione sono stati sicuramente la libertà che viene lasciata agli ingegneri del software all'interno dell'azienda, il fatto che non venga data troppa importanza al tipo di linguaggio usato, o alla quantità di linguaggi di programmazione conosciuti, ma piuttosto si dà peso alle competenze generali e alla capacità di logica e ragionamento, in quanto l'ingegnere del software non è una figura orientata unicamente alla programmazione, bensì deve pensare anche al project management e deve essere una figura pronta ad adattarsi alle necessità dei singoli progetti.
In questo seminario, quindi, non ci si è concentrati su alcun metodo di sviluppo software specifico.

U-HOPPER

U-Hopper è un'azienda trentina che si occupa di aiutare le aziende ad analizzare i loro big data per estrarre da essi soluzioni legate all'intelligenza artificiale che possano migliorare i processi aziendali.
Anche in questo ambito vengono utilizzate tecniche dell'ingegneria del software, nello specifico U-Hopper adotta un approccio agile: viene diviso l'intero progetto in sottoparti, dette Milestones, ognuna di esse è divisa in Issues e sulla base di esse viene generalmente stimata la deadline di consegna.
Per quanto riguarda la parte pratica: prima viene scritto il codice, poi viene testato e infine viene fatta la review.
Un'altra procedura che l'azienda U-Hopper ha preso dall'ingegneria del software e segue scrupolosamente è la stesura della documentazione, un passaggio fondamentale per fare in modo che il progetto possa essere seguito facilmente da un vasto e mutabile team di persone in un lungo periodo di tempo.
Essa serve, quindi, a far sì che chiunque si approcci al progetto, in qualsiasi momento, leggendo la documentazione, capisca velocemente di cosa si occupa, ad esempio, la nostra applicazione, e di come è strutturata.

REDHAT

Questo seminario è stato tenuto da Mario Fusco, lo sviluppatore della libreria open source lambdaj per Java, ed è stato strutturato come una presentazione dei pro dell'Open Source.
Il mondo open Source, per come ci è stato presentato consente:

- *la condivisione della conoscenza*. Possiamo imparare, in primis, vedendo come è stato scritto il codice altrui, inoltre chiunque può migliorare il codice che noi pubblichiamo, in questo modo si può arrivare più velocemente ad una soluzione migliore. Inoltre, allo stesso tempo, si può imparare dai propri errori.
- una forte *meritocrazia* e *possibilità di visibilità* perchè la persona è legata alle proprie idee e i propri lavori che sono esposti pubblicamente
- trasparenza sui sistemi implementati

MICROSOFT

Questo seminario è stato fortemente incentrato su una singola parte dell'ingegneria del software, ovvero il software testing.
Il testing è una verifica, manuale o automatizzata, dei requisiti funzionali e non del software che sto realizzando.
Il testing può essere fatto su diversi livelli e di conseguenza avremo diverse tipologie di testing, un esempio è l'acceptance testing. Come ci è stato spiegato dal relatore, Diego Colombo, in un'ottica lavorativa e soprattutto in una modalità di lavoro agile, conviene decidere e rendere chiare fin dall'inizio le condizioni che deve soddisfare il nostro software ad ogni consegna, in modo che il cliente sia vincolato nelle richieste in corso d'opera.

Ci è stata, quindi, esposta l'importanza di questa fase del ciclo di vita del software, infatti, essa serve a prevenire bug, problemi e regressioni, a controllare che il proprio codice sia conforme alle specifiche richieste e ad avere un feedback sul design.

MOLINARI

Il seminario tenuto dal professor Andrea Molinari si è incentrato sui sistemi legacy, ovvero quei sistemi informativi formati interamente o in parte da componenti obsolete, indistintamente che facciano esse parte dell'hardware, del software, delle apparecchiature di rete, dei processi o delle persone.
È stato importante fare una panoramica sui sistemi legacy in quanto, nonostante l'obsolescenza, quest'ultimi sono sistemi tuttora persistenti e fortemente utilizzati, quindi un ingegnere del software deve tenere conto della loro esistenza quando va a sviluppare un sistema, nel caso in cui quest'ultimo debba interfacciarsi e interagire con un sistema legacy.
Alcune caratteristiche distintive dei sistemi legacy sono:

- hardware mainframe, datato, di grosse dimensioni e fortemente dispendioso a livello energetico
- effetto lock-in con fornitori
- personale addetto ridotto
- generalmente vengono usati Sistemi Operativi realizzati ad hoc sulle strutture hardware

La convivenza tra sistemi legacy e sistemi informativi moderni è consentita da 4 tecniche di approccio:

- dismissione: il sistema viene smantellato e le procedure vengono affidate a nuovi software
- migrazione: i dati e i software vengono portati nel nuovo ambiente per dimettere poi il sistema legacy
- interazione: il sistema legacy continua a persistere e interagisce con il nuovo sistema
- inclusione: il sistema vecchio viene incapsulato nel nuovo sistema informativo

Anche a livello di sistemi legacy è stata ribadita l'importanza dell'ingegneria del software, specialmente per quanto riguarda l'uso di diagrammi UML. Questo linguaggio di modellazione risulta quasi sottodimensionato e per questo si fa particolarmente uso delle sue versioni più evolute, come ad esempio SysML, che consente la realizzazione di due nuovi tipi di diagrammi, quali il "parametric diagram" e il "requirement diagram".
Al contrario della maggior parte dei seminari, dove venivano presentati in maniera particolare gli aspetti positivi dell'approccio agile, in questo caso i progetti di grosse dimensioni, i team numerosi e fortemente decentrati scoraggiano l'uso della suddetta metodologia e favoriscono, piuttosto, metodologie predittive, iterative e incrementali.

MARSIGLIA

Questo seminario, tenuto da Gerardo Marsiglia, si è incentrato su due temi principali:

- il ciclo di vita dei software, con i passaggi canonici che ci sono sempre stati presentati: pianificazione, definizione dei requisiti e definizione del design, sviluppo dell'app mediante la stesura del codice, fase di testing, distribuzione dell'applicativo e infine il suo continuo mantenimento.
- il processo di modernizzazione delle applicazioni, necessario al fine di sfruttare tutti i vantaggi delle nuove architetture software.

Sono state presentate alcune professioni legate all'ingegneria del software (architetto, consulente, ingegnere, manager), inoltre è stato fatto un excursus sulle varie metodologie di approccio allo sviluppo del software partendo dalle tecniche tradizionali come il processo a cascata, passando poi per il metodo iterativo, agile e infine DevOps.

APSS - TRENTINO

Quest'ultimo seminario era volto a spiegare come viene organizzato il dipartimento informatico e come viene gestita l'importante mole di dati di una grossa realtà come quella dell'Azienda Provinciale dei Servizi Sanitari di Trento.
Il tutto viene suddiviso in 4 macroaree:

- le infrastrutture, ovvero tutta la struttura hardware volta alla comunicazione e gestione dei dati
- le applicazioni, ovvero i software sia clinici che amministrativi
- il data management
- l'ingegneria clinica, le apparecchiature mediche effettive

È stato evidenziato come nell'ambito sanitario l'approccio agile non sia consigliato, a causa dell'uso degli MVP, Minimum Viable Product, in quanto non può essere distribuito nell'area medica un prodotto parzialmente funzionante, perché il tasso di errore deve essere pari a zero, lo standard dell'MVP si alza quindi notevolmente, andando a perdere il significato effettivo e lo scopo di un MVP ovvero la realizzazione di un prodotto da mettere nel mercato nel minor periodo possibile, nell'ottica di andarlo a migliorare successivamente.

IL NOSTRO APPROCCIO

Il nostro gruppo ha cercato di approcciarsi al progetto seguendo le fasi principali e caratteristiche del ciclo di vita di un software, presentateci ad esempio da Gerardo Marsiglia e durante tutta la durata del corso.
Per come è stato impostato il corso ci è venuto spontaneo lavorare principalmente seguendo il metodo a cascata, piuttosto che il metodo agile, maggiormente favorito nel seminario di Kanban e da aziende come IBM e U-Hopper.
Nel concreto, abbiamo realizzato in maniera consecutiva i singoli deliverables che di fatto coincidevano con gli step di realizzazione dell'applicativo basandoci, di volta in volta, sul prodotto realizzato fino a quel momento, per poi progredire nello sviluppo della web app, andando a fare successivamente eventuali modifiche che permettessero di mantenere coerente, puntuale e aggiornata l'intera documentazione.
Abbiamo, infatti, imparato l'utilità di una buona documentazione, opinione fortemente sostenuta sia dal professor Molinari, sia da Daniele Miorandi di U-Hopper, in quanto essa ci ha permesso di orientarci facilmente nel nostro stesso progetto anche dopo periodi di inattività.
In generale abbiamo notato una forte somiglianza con gli strumenti utilizzati, ad esempio diagrammi UML e realizzazione di mock-up per la user interface, come visto in Blue Tensor o come fortemente consigliato dal professor Molinari, oppure l'uso di GitHub come in U-Hopper.

ORGANIZZAZIONE DEL LAVORO

Per la maggior parte dei casi ogni deliverable è stato suddiviso in sottoparti e ognuna di esse poteva essere ulteriormente porzionata. In questo modo siamo riusciti a ripartire le varie sezioni tra noi membri del gruppo, in modo da evitare al massimo collisioni e sovrapposizioni di mansioni.
I vari compiti sono stati assegnati in base alle competenze e alle preferenze di ciascun componente.
Le varie interazioni tra componenti sono avvenute in forma più frequente tramite lo scambio di messaggi, a cadenza più irregolare con degli incontri fisici e talvolta con delle chiamate o incontri virtuali.
L'utilizzo di GitHub è stato limitato e confinato principalmente alla parte della scrittura del codice, per quanto riguarda i file di documentazione, invece, sono stati realizzati congiuntamente tramite l'uso di Google Document.
Alcuni strumenti utilizzati come supporto sono stati:

- *Google* Sheets: per tenere conto del monte ore
- *Figma*: per la realizzazione dei mockup
- *LucidChart*: per la realizzazione dei vari diagrammi

RUOLI E ATTIVITÀ

Componente del team	Ruolo	Principali Attività
Riccardo Miolato	Team leader e sviluppatore	Si è occupato dell'organizzazione del team, del proporre soluzioni per realizzare al meglio l'idea iniziale e del mettere in relazione i vari membri del gruppo per superare le difficoltà. Per quanto riguarda lo sviluppo della webapp ha dato il contributo più grande nella parte di back-end.
Vladislav Bodrug	Sviluppatore e analista	Si è occupato, in quanto membro con esperienza pregressa, di proporre soluzioni alternative per il superamento degli ostacoli incontrati dal team. Grazie alla sua esperienza nello sviluppo di interfacce utente ha voluto dare un contributo maggiore in questo ambito anche nella nostra webapp.
Giada Vicentini	Sviluppatrice e analista	Nonostante fosse il membro del gruppo con meno esperienza, avendo un background diverso, il suo ruolo non è stato per niente marginale. Il contributo maggiore che ha portato nel gruppo è stato un punto di vista diverso, utile per guardare le cose con un'altra prospettiva e molta buona volontà nell'imparare delle nuove tecnologie per aiutare sia nel front-end, sia nel back-end.

CARICO E DISTRIBUZIONE DEL LAVORO

Dal foglio di calcolo realizzato risulta il seguente monte ore per ciascun componente del gruppo, suddiviso inoltre nei vari deliverables.
È possibile notare una buona omogeneità nella distribuzione di ore nei primi tre deliverables in quanto si è cercato di suddividere il lavoro, mentre per quanto riguarda gli ultimi due c'è stata una divisione più netta, in quanto, proprio a causa delle maggiori capacità nell'ambito della programmazione Riccardo Miolato e Vladislav Bodrug si sono suddivisi il d4, rispettivamente curando la parte di back-end e front-end, mentre Giada Vicentini si è occupata principalmente della stesura del d5.

	D1	D2	D3	D4	D5	TOT
Riccardo Miolato	24	15,5	9	80	3	130.5
Vladislav Bodrug	15	15	5,5	85	5	125.5
Giada Vicentini	14	12	7	43	19	95

CRITICITÀ

La prima criticità incontrata è stata quella della suddivisione dei compiti, in quanto si tendeva a fare tutto insieme, ci si è però presto accorti che ciò implicava impiegare molto più tempo per ogni singola parte e si è suddiviso ogni deliverable in sottoparti come spiegato sopra nella sezione “Organizzazione del lavoro”.
Un’altra criticità presente durante quasi la totalità del progetto è stata la comunicazione. Spesso, infatti, per comodità i membri tendevano a parlare singolarmente con gli altri componenti del gruppo, così di volta in volta il terzo non prendeva parte a scelte o conversazioni. Nonostante le suddette questioni non fossero punti focali del progetto, questo metodo di approccio e risoluzione ha in qualche modo allentato la coesione del team, ma nel complesso non ha influito in maniera eccessiva .

AUTOVALUTAZIONE

Complessivamente, tutti i membri del gruppo hanno dimostrato impegno costante durante l'intero progetto, contribuendo ciascuno con le proprie competenze e capacità. Questo ha generato una distribuzione del lavoro molto variegata, soprattutto nei due ultimi deliverables. Al contrario, la fase iniziale di ideazione e progettazione del software ha visto la collaborazione di tutti, consentendoci di acquisire una comprensione approfondita delle diverse metodologie di approccio a un progetto di ingegneria del software e delle loro differenze.
Riccardo Miolato è stato sicuramente il componente più attivo, soprattutto a livello di ore dedicate al progetto. Vladislav Bodrug è sempre risultato puntuale nell’eseguire le consegne assegnategli. Infine, Giada Vicentini si è sempre dimostrata disponibile ad aiutare chi ne avesse bisogno e portare le sue idee all’interno del gruppo.

Sulla base delle suddette considerazioni e la qualità del lavoro eseguito e del prodotto finale ottenuto, la nostra autovalutazione è la seguente:

	voto
Riccardo Miolato	30
Vladislav Bodrug	29
Giada Vicentini	28