UNIVERSITÀ
DI TRENTO

Dipartimento di ingegneria e scienza dell'informazione

Progetto:

Vocable

Titolo del documento:

Analisi del lavoro e Autovalutazione

Document info:

Document info

Doc. Name	D5-Vocable_Autovalutazione	Doc. Number	D5
Description	Il documento contiene una analisi del lavoro complessiva e autovalutazione del gruppo		

INDICE

Scopo del documento:

In questo documento ci dedicheremo inizialmente all'analisi di vari approcci all'ingegneria del software, illustrati da professionisti provenienti da diverse aziende e realtà lavorative. Questi esperti hanno condiviso con noi le loro esperienze personali durante una serie di seminari. Successivamente, descriveremo l'approccio adottato per lo sviluppo di questo progetto, approfondendo l'organizzazione del lavoro, la suddivisione dei ruoli all'interno del team, la distribuzione del carico di lavoro tra i membri e le criticità emerse lungo il percorso. Concluderemo con una riflessione finale che includerà una breve autovalutazione del progetto.

Approcci all'ingegneria del software:

In questo paragrafo offriremo una sintesi concisa dei contenuti di ciascun seminario, evidenziando in particolare i vari approcci all'ingegneria del software. Concluderemo con una descrizione del nostro metodo di lavoro, mettendo in luce le somiglianze e le differenze rispetto alle esperienze che ci sono state illustrate nel corso del semestre.

BlueTensor (Azienda AI)

L'approccio di ingegneria del software dell'azienda BlueTensor è molto simile a quello di tipica ingegneria del software che discutiamo a lezione, ma differisce in quanto l'azienda si occupa di intelligenza artificiale e delle sue applicazioni pratiche. L'azienda crea prodotti che spesso non sono compresi al 100% dai propri clienti al momento della richiesta, quindi parte del lavoro che fanno è comunicare con i propri clienti anche per cercare di capire quale prodotto cercano e la fattibilità del prodotto. Il processo parte con una analisi del problema e della fattibilità di implementare una soluzione, un design di un proof of concept per presentare il prodotto, una preparazione dei dati, un minimum viable product seguito dal training del programma e infine un rilascio di un software che è in miglioramento continuo, analizzando nuovi dati il software è in grado di essere meglio ogni volta.
I punti di forza del loro approccio sono sicuramente la quasi totale sovrapposizione del prodotto finale al prodotto che un cliente si aspetta, perché fanno molta attenzione a cercare di comunicare al cliente cosa vuole e come funzionerebbe un potenziale sistema. Il punto di debolezza del loro approccio è legato semplicemente al fatto che i loro sistemi più specifici hanno bisogno di grandi quantità di dati per essere addestrati, in quanto intelligenze artificiali.

IBM (Cloud Computing)

IBM è uno dei maggiori fornitori di cloud computing nel settore.
Il servizio di cloud computing di IBM offre ai propri clienti software e hardware per creare un servizio online. Il loro tipico cliente può ordinare hardware più o meno complesso per diminuire il carico di lavoro impiegato per implementare le infrastrutture o avere più controllo, da Serverless Code Engine a intere macchine fisiche. Il secondo passo è capire quale tipo di software usare per le funzionalità richieste e per garantire un livello di sicurezza adatto, per poi sviluppare l'applicazione. Il punto di forza dello sviluppo di software attraverso il cloud IBM è l'accessibilità, che il cliente sia inesperto o molto esperto ci sono soluzioni per tutti. Il punto di debolezza è il fatto che il funzionamento del proprio prodotto sia dipendente da una azienda esterna, se IBM dovesse avere problemi a livello fisico con i propri server anche l'applicazione del cliente smetterebbe di funzionare.

META (Lavorare in una grande azienda)

In questa conferenza ci è stato spiegato come sia internamente organizzato lo sviluppo di software dentro META, una delle più grandi aziende di social network.
L'approccio di meta è quello di sviluppo totale degli strumenti usati attorno al prodotto, i programmi che vengono usati all'interno dell'azienda sono quasi tutti open-source e modificati dall'azienda stessa per incontrare i propri bisogni.
All'interno di META ogni dipendente è incentivato dai suoi superiori a provare tipi diversi di mansioni, in modo da poter fare esperienza in vari campi, diventando quindi esperti in più tipi di programmi e stili di sviluppo.
Il punto di forza di questo tipo di approccio è il miglioramento di ogni singolo developer all'interno dell'azienda: ognuno col tempo cresce e impara nuovi modi di pensare all'ingegneria del software.

u-hopper (Git)

In questa conferenza abbiamo scoperto come l'azienda u-hopper usi un approccio agile allo sviluppo.
Il loro approccio è basato sul definire delle milestone, dividerle in piccole issues e calcolare una deadline complessiva in base al costo complessivo stimato delle varie issues. Per lo sviluppo, l'azienda utilizza tutte le funzionalità di un software simile a GitHub ma con la possibilità di creare file più grandi.
L'azienda lavora in parallelo su più porzioni di codice in ambienti separati in modo da poter controllare l'effettivo funzionamento del codice ad ogni stadio del development.
All'interno di questa azienda vengono spesso utilizzate applicazioni monolitiche invece che modularizzare le applicazioni in microservizi.
Il grande punto di debolezza del loro approccio è la complessità dello sviluppo di queste applicazioni monolitiche, che richiedono molto tempo e molte risorse per essere sviluppate.
I punti di forza del loro approccio sono la grande organizzazione dello sviluppo e la riduzione dell'overhead, che con una modularizzazione del software diventerebbe molto grande.

RedHat (Open-Source)

La conferenza di RedHat ci ha insegnato l'approccio open-source alla programmazione, la possibilità di imparare a crescere insieme a persone molto più esperte che possono darti un'idea di quanto il tuo codice sia sbagliato e come migliorarlo. L'approccio di Fusco è fantastico, ma funziona solo per programmatori che hanno abbastanza maturità da non prendere sul personale le critiche degli altri sul proprio codice, come anche lui ha detto bisogna imparare ad accettare che il proprio codice non è il migliore possibile.

Microsoft (Software testing)

Questa conferenza si è incentrata sul cosa vuol dire fare testing e perché sia importante fare development anche tenendo a mente quali potenziali problemi potrebbe dare il codice, test driven development invece che design driven development. L'approccio ha il vantaggio di essere molto efficiente dal punto di vista del tempo guadagnato in testing, perché sviluppare un programma attorno a cosa non dovrebbe fare oltre a cosa dovrebbe fare rende la fase di testing molto più veloce: il codice si rompe oggettivamente di meno.

Molinari (Legacy Systems)

Il professor Molinari ci spiega perché una grande parte del sistema finanziario sia basato su sistemi legacy, dagli ovvi contro quali il consumo di tanta energia e i costi di manutenzione ai non tanto ovvi pro quali la potenza di calcolo e il non dover investire ulteriore capitale per un upgrade. Il mondo legacy per fortuna non ha niente in comune con il lavoro del nostro gruppo, non diventeremo sicuramente developer cobol, ma ci insegna una grande lezione: dobbiamo continuare a imparare. In un mondo che continua a innovare noi dovremo cercare di rimanere al passo e non finire ancorati alla stabilità di un sistema che prima o poi diventerà obsoleto.ù

Marsiglia (Software Lifecycle)

L'architetto del software Marsiglia ci ha parlato in questa conferenza di come funzionano i software moderni, software fatti per minimizzare il downtime e permettere supporto a lungo termine. Il lato positivo di questo approccio è sicuramente che funziona, il software implementa sia i requisiti funzionali che non funzionali, inoltre è un software in continua evoluzione: Il fatto che il software venga continuamente supportato permette ai developers di aggiungere funzioni, rendere il codice più efficiente e soprattutto di aggiungere caratteristiche moderne. Il lato negativo di questo tipo di software è che per permettere il supporto continuo viene a costare molte risorse, una situazione dove però i pro mettono in ombra i contro visto che un software che funziona permette alle aziende di tenersi al passo e continuare a guadagnare più delle risorse richieste per lo sviluppo.

APSS & Trentino.ai (Sanità e Intelligenza artificiale)

In questa conferenza abbiamo imparato come funziona il budgeting in una azienda di larga scala come l'APSS e come l'intelligenza artificiale sta cambiando anche questo mondo. Abbiamo visto come l'azienda sanitaria trentina si rapporta alla tecnologia nello specifico, quali tipi di sistemi usa e la enorme scala alla quale vengono usati, abbiamo anche capito come le applicazioni di IA usate dal sistema sanitario trentino possano anonimizzare ad esempio i dati dei pazienti affetti da retinopatia per poter ricavarne delle statistiche senza rischiare di intaccare la loro privacy.

IL NOSTRO APPROCCIO

Abbiamo fatto tesoro delle conferenze, tenendo in particolare a cuore alcuni degli approcci, in particolare abbiamo preso spunto dall'approccio agile di u-hopper e abbiamo utilizzato in piccolo il test driven development come spiegato da microsoft per sviluppare api che non presentino problemi.
Siamo riusciti a ottenere conoscenze specifiche senza toglierci la possibilità di avere una visione dell'insieme e fare di tutto. Come nell'azienda META, ci siamo sentiti spronati tra di noi a imparare tutto il possibile in ogni campo, dandoci una mano a vicenda quando necessario.

Organizzazione del lavoro:

Nel primo deliverable abbiamo riscontrato una certa disorganizzazione, dovuta alla scelta di una persona non adatta come project leader. Dopo aver valutato il lavoro delle prime settimane, è emerso che Anita Scortegagna sarebbe particolarmente adatta per ricoprire questo ruolo. Abbiamo quindi deciso all'unanimità di nominarla come nuova project leader.
Abbiamo mirato a una suddivisione equilibrata dei compiti, cercando di distribuire le attività in modo equo tra i membri del team, tenendo conto anche delle competenze dei singoli membri e delle loro preferenze personali.
Per ogni documento, abbiamo elaborato delle scalette di lavoro per migliorare l'organizzazione e mantenere alta la motivazione del team.
Le comunicazioni tra i membri del gruppo sono avvenute principalmente tramite app di messaggistica e incontri di persona. Quando non era possibile riunirci fisicamente, abbiamo optato per chiamate online, durante le quali condividevamo i progressi e discutevamo le problematiche.
Come strumentazione, ci siamo affidati all'uso di *GitHub*, che è stato cruciale per la creazione e condivisione del codice. Inoltre, abbiamo utilizzato *Google Documents* per facilitare la modifica parallela dei documenti, consentendoci di monitorare le modifiche in tempo reale. I vari diagrammi, sono stati realizzati tramite *LucidChart* e i mockup del sito tramite *Figma*.

Ruoli e attività:

In questa tabella riassumiamo il ruolo di ciascun membro del team nel progetto. In alcuni casi, abbiamo avuto una suddivisione ben definita dei compiti, mentre in altri abbiamo collaborato collettivamente sugli stessi aspetti.

Componente del gruppo	Ruolo	Principali attività.
Anita Scortegagna	Project Leader, Creatrice del primo Mockup, Analista dei requisiti non Funzionali, Responsabile OCL, Co-responsabile Class diagram, Esperta di Front-End e Deployment, Co-responsabile API.	Anita è stata la mente dietro il progetto, ci ha aiutato a capire cosa fare e come dividere i compiti. Ha creato il primo mockup e si è occupata molto dei requisiti non funzionali in D1 e D2, occupandosi in particolare del class diagram nel D3. Si è occupata principalmente del Front-End, deployment e delle API.
Pietro Mazzocco	Responsabile diagramma dei componenti, Requisiti funzionali con use case, Controllo errori nei deliverable, Esperto debugging e Database, Co-responsabile API.	Pietro M. si è principalmente dedicato ad aiutare a sistemare il D1 ed aggiungere le prime cose mancanti, ha aiutato principalmente nel D2 e D5 e si è occupato molto dell'ordine dietro ai file. Nella fase di development si è principalmente occupato di Database, debugging ed API.
Pietro Tabladini	Co-responsabile Class diagram, Analista dei requisiti Funzionali, Responsabile context Diagram, Esperto Front-End e Testing, Co-responsabile API.	Pietro T. Ha lavorato principalmente al D1, D2 e D3 dedicandosi soprattutto al passaggio dal diagramma di contesto a quello delle classi e della stesura del diagramma di contesto. Durante il development si è occupato principalmente di Front-End, testing ed API.

Carico e distribuzione del lavoro:

Abbiamo tenuto conto delle varie ore di lavoro per ciascuna parte del progetto tramite un foglio di calcolo. Come si può notare il lavoro è stato distribuito in modo abbastanza omogeneo, in particolare nei primi tre deliverables. Nel quarto invece, si può notare che le ore sono più disomogenee,
Abbiamo monitorato le ore di lavoro per ciascuna parte del progetto utilizzando un foglio di calcolo. Come si può notare, la distribuzione del lavoro è stata piuttosto equilibrata, in particolare nei primi tre deliverables. Tuttavia, nel quarto deliverable, le ore di lavoro risultano meno uniformi. Infatti, Pietro Mazzocco ha avuto un coinvolgimento leggermente inferiore rispetto agli altri membri del team a causa di problemi di salute. Per il resto, il lavoro è stato distribuito in modo abbastanza equo.

Di seguito è riportata la tabella che mostra le ore spese da ciascun membro del gruppo per ogni documento:

	D1	D2	D3	D4	D5	Totale
Anita Scortegagna	10	12	12.6	96	1.5	132.1
Pietro Mazzocco	9	12	16.3	77.5	3	117.8
Pietro Tabladini	10	11.5	13.5	83.3	0.5	118.8
Totale:	**29**	**35.5**	**42.4**	**256.8**	**4**	**368.7**

Criticità:

Le criticità che abbiamo affrontato sono state poche e per la maggior parte legate a problemi personali o alla sovrapposizione delle tempistiche con lavori di altri corsi. Durante la stesura del primo documento, stavamo ancora cercando di capire come suddividere il lavoro. Ognuno cercava di fare qualcosa ma c'era un chiaro bisogno di capire i ruoli. Durante il *D2* i ruoli sono emersi quasi in modo spontaneo, era evidente quali fossero gli interessi di ognuno di noi e da allora abbiamo avuto pochi problemi legati a quello.
Verso la fine del progetto, ovvero dal *D4* in poi, abbiamo avuto un problema legato alla sovrapposizione delle tempistiche del corso *"Algoritmi e strutture dati"* con la scadenza finale di questo progetto. Questo ci ha fatto affrontare un carico di lavoro particolarmente intenso, influenzando il nostro impegno complessivo, finendo per avere una generale disorganizzazione che ci ha portato a rimandare fino a settembre la consegna del progetto, data la disponibilità maggiore di ognuno di noi verso luglio e agosto.

Autovalutazione:

Per quanto nessun membro del team sia sempre riuscito ad essere sempre presente in tutte le fasi del progetto, siamo riusciti a mantenere sempre il conto delle ore di lavoro compiute da ognuno, portando a un carico di ore bilanciato adeguatamente tra di noi. Siamo fieri di tutta la strada che siamo riusciti a fare insieme e pensiamo di meritare una valutazione che rifletta la qualità e quantità del lavoro fatto.

Studente	Voto:
Anita Scortegagna	30
Pietro Mazzocco	30
Pietro Tabladini	30