Documento:**D5-G09**
Revisione: **1.0**

Money Expense

Report finale del progetto

Scopo del documento

In questo documento viene presentato il report conclusivo del progetto portato avanti dal gruppo **G09** durante il corso di Ingegneria del Software 2023/2024.
Il documento inizia con una serie di brevi descrizioni e riflessioni relative ai vari seminari seguiti durante il corso.
Segue la parte in cui viene articolato come il lavoro durante il corso è stato diviso ed organizzato tra i membri del team.
A seguire, presentiamo il tempo che ogni singolo componente del team ha dedicato allo sviluppo del progetto.
In quanto le ore di lavoro sono state suddivise in base alle attività, sarà quindi possibile evincere chi ha lavorato maggiormente a quale documento e le eventuali motivazioni.
Verrà introdotto successivamente un elenco delle criticità riscontrate ed eventualmente i mezzi utilizzati per risolvere.
In conclusione, presenteremo un'autovalutazione indicativa relativamente al lavoro svolto sia come gruppo che come singolo membro.

Approcci all'ingegneria del Software

Bluetensor
In questo primo seminario, l'azienda ha parlato di come loro sviluppino le loro varie soluzioni in ambito intelligenza artificiale. Hanno spiegato la loro giornata tipo e come strutturano il loro lavoro.
In particolare, loro utilizzano un approccio molto orientato al Agile, in quanto ogni progetto è unico e assestante, e ha bisogno di un forte feedback loop con il cliente.
Per questo, loro si focalizzano molto sul partire da uno studio di fattibilità, necessario per evitare grandi perdite di tempo, e progetti fallimentari. Dopo di che si redige un documento molto dettagliato con tutto ciò che ci si pone come obiettivo di realizzare, per passare infine all'implementazione effettiva della soluzione richiesta. Durante la fase di sviluppo, ci sono molti rilasci, per permettere al cliente di capire se ciò che sta venendo sviluppato è ciò che si aspetta, ed eventualmente è ciò che si aspettava. Ci è molto piaciuto questo approccio perché permette di trovare in fretta incongruenze tra la richiesta e l'offerta, limarle e raggiungere il risultato desiderato dallo stakeholder. Questa flessibilità non va in contrapposizione, nonostante tutto, con il forte lavoro fatto a priori, per essere sicuri che ci siano i prerequisiti necessari al partire, ed una solida base concordata. Ci è sembrato un ottimo connubio tra praticità e formalità.

Laboratorio Kanban
In questo seminario ci è stata proposta una metodologia di ingegneria del software molto particolare e diffusa, che è il metodo Kanban. Consiste in una pool di micro task, che possono venire assegnate, messe in pausa e completate dai vari addetti ai lavori. Ognuna di queste task è disposta in una sorta di lavagna divisa in compartimenti, dove compaiono quelle non ancora assegnate, quelle alle quali si sta lavorando, e quelle da finire. Ogni singolo lavoratore ha un numero di task a cui può lavorare, e che varia in base a una formula molto definita, per permettere il ricircolo e una produttività dimostrata essere maggiore. E' parso a noi una metodologia molto interessante. Il fatto che fosse artigianale, nel senso con biglietti fisici, e la possibilità di segnare le cose come completate, può aumentare la soddisfazione, e per questo può avere maggior successo. Inoltre è provato che questo metodo aumenti la produttività di chi ne fa uso. Unico problema che ci è parso è che sembra molto compartimentato, e lascia minor flessibilità, ma forse è solo una percezione esterna, che non si ha una volta che si utilizza attivamente la metodologia specifica.

IBM
Nel secondo seminario, è venuta IBM. Sicuramente una azienda diversa, in quanto longeva e di dimensioni di gran lunga superiori alla precedente. In particolare, il focus di questo seminario era il dipartimento Cloud, e le soluzioni che questi sono in grado di creare. Ci è stato mostrato un esempio di progetto possibile ed una demo dello stesso. Qui sono apparse le prime differenze con BlueTensor, in quanto ci è stato detto come generalmente questi progetti sono più simili ad una metodologia waterfall. Questo perché spesso chi ha bisogno ha già degli studi di fattibilità e idee più chiare, oppure

sono in grado di gestire l'utilizzo del Cloud senza assistenza diretta dell'azienda. Ci siamo quindi resi conto che non esiste una metodologia perfetta per tutti, ma esiste una metodologia ottimale per ogni caso. Ad ogni modo, anche IBM fa uso estensivo di documenti, artefatti e diagrammi, molti anche sviluppati con tool interni, per rappresentare i propri progetti, dando prova di quanto sia utile avere degli oggetti fisici che testimoniano l'obiettivo comune. Questo approccio, tutto sommato ci è parso funzionale, ma che lascia poco spazio ad eventuali errori, e quindi prono a più problemi possibili rispetto al seminario precedente.

Meta

Meta ha presentato l'azienda e le metodologie del software utilizzate. La presentazione è cominciata dicendo che a Meta sono usati una marea di linguaggi, e che quindi non è tanto importante la tecnologia usata in sé, ma il software risultante e la sua qualità. Il relatore ci ha riferito che in Meta non si usa una metodologia specifica. Lui la ha chiamata di buon senso, che si avvicina all'Agile, ma è molto più artigianale, e cucito addosso ai singoli team. Questo per dare massima libertà di espressione a tutti i vari contribuenti al progetto e anche per evitare burocrazia non necessaria. Ci sono tuttavia comunque documenti condivisi e canali di comunicazione dedicati, in quanto vitali per evitare di lavorare a multipli progetti diversi quando si dovrebbe lavorare ad uno insieme. Questo approccio ci è sembrato estremamente snello e libero. Ci è parso che se tutti i lavoratori sono seri, questo può portare al miglior risultato, ma se qualche parte fosse un pò meno propensa allo spendersi, uno standard più strutturato potrebbe aiutare al successo del progetto.

U-Hopper

In questo seminario, ci e ' stato parlato di una azienda più locale. Nonostante ciò è stato molto interessante. Dopo una introduzione sulla compagnia, le tecnologie e i linguaggi, si è parlato di metodologie di ingegneria del software. In particolare, ci è stato detto di come loro lavorino per sprint, seguendo quindi un approccio molto agile, veloce e continuo. Ogni due settimane si lavora a micro task, che vanno a comporre un macro progetto finale. E' stato posto il focus su come viene usato Git e i branch per fare tutto ciò e di come sia strutturato questo lavoro per poter dare la miglior struttura possibile al progetto risultante. Essendo anche questa una azienda di fatto di consulenza, e lavorando a progetti, è necessario un forte feedback loop sia interno che esterno, quindi con il cliente, che è fortemente integrato in tutte queste meccaniche. Dopo di che, anche qui ci sono stati descritti i documenti e gli artefatti prodotti. Sono stati dettagliati i test, la documentazione del progetto iniziale e durante lo sviluppo, l'importanza delle review e il deployment del progetto completato. Anche qui, il processo è sembrato molto articolato. Ci ha lasciato un pò perplessi l'enorme quantità di artefatti e burocrazia inclusa in tutto il processo, che ci è sembrata eccessiva per l'azienda, ma poi da come è stato detto dallo stesso relatore, visti i clienti esterni, spesso non esperti di tecnologia, è necessario. Per ciò, alla fine sembrava essere la soluzione migliore per il team e alla tipologia di progetti che doveva affrontare. E' stato anche posto un focus sui tool, che sono condivisi da tutti, in modo da avere dei workflow comuni e di avere possibilità di supporto reciproco, che è fondamentale in un team.

RedHat
Questo seminario ha avuto un forte focus soprattutto per quanto riguarda il mondo open source più che il mondo dell'ingegneria in sé, ma ha avuto vari spunti anche su questa. Oltre a spiegare cosa è l'open source e come si può guadagnare, è stato raccontato come si sviluppa un progetto open source. Di fatto, visto che i contributori sono molti, molto geograficamente sparsi, e di fatto quasi mai pagati per contribuire, per fare in modo che un progetto di questa tipologia abbia successo è necessario che ci sia una documentazione molto ben strutturata, comune e facilmente leggibile. Inoltre, è necessario che ogni task sia ben documentata e che si sappia a che punto si è arrivati ognuna di esse. E' necessario quindi che se uno ha incontrato e risolto un problema, lo renda noto, che cosi' futuri contributori sapranno di questa soluzione senza dover perdere troppo tempo a cercare loro un differente approccio. Le metodologie che sono state esposte sono sembrate molto burocratizzate, e non facilmente applicabili ad un team come quello di lavoro per il progetto da noi sviluppato, ma adatte a progetti del tipo open source. Va anche precisato che alcuni progetti sono seguiti da vere e proprie aziende, e che quindi non tutto è aperto, oppure qualche decisione è più centralizzata. Per questo ogni caso va valutato singolarmente in questo mondo.

Microsoft
Il focus di questo seminario è stato il testing di un software e la sua importanza, oltre che la sua varietà. In Microsoft, in quanto grande azienda, è necessario poter garantire che i servizi siano testati, sufficientemente robusti per i traffici che attirano e per la loro importanza. Il testing è quindi parte integrante e fondamentale dell'ingegneria del software, per garantire che il progetto abbia un successo misurabile. È stato inoltre accennato che anche in Microsoft ci sono varie metodologie, che variano a seconda del team, e quindi del tipo di prodotto che si ha per le mani, ma generalmente sono tutte alternative della metodologia Agile, divisa in sprint. Ci è parso molto sensato visto le dimensioni dell'azienda e il numero di stakeholder inclusi nei vari progetti. Unico piccolo svantaggio forse la burocrazia non indifferente, che ti fa passare più tempo a scrivere documenti che codice effettivo, ma è necessario per poter coordinare tanta forza lavoro per progetti di simili dimensioni.

Molinari
Abbiamo avuto in questo seminario una idea di come sia lavorare con sistemi legacy, e di quanti siano ancora rilevanti nel mondo odierno. Nonostante ciò, sono sistemi non facili da trattare, in quanto molte tecnologie usate non sono più allo stato dell'arte, e molti sviluppatori per questi linguaggi non sono più in attività. Inoltre è un lavoro sempre di rinnovamento, e quasi mai di innovazione. Per questo, dal punto di vista dell'ingegneria del software, lavorare con sistemi legacy è diverso dal progetto che stiamo sviluppando, in quanto Agile non è veramente applicabile, come ci ha riferito il relatore. Spesso le figure professionali sono molto distinte, spesso essere sincronizzati non ha utilità, in quanto si agisce su un componente, che non va a mutare il sistema. Spesso i sistemi hanno delle documentazioni enormi. Per questo le metodologie si adattano molto al contesto e al progetto in questione, e non sono mai uniche. Tendenzialmente, per lavorare in questi sistemi, il tempo richiesto è maggiore. Una costante però tra il mondo dello sviluppo più classico e quello legacy, è il forte utilizzo dei diagrammi UML come

documentazione, rimarcando la loro importanza. Non abbiamo potuto sperimentare veramente queste metodologie durante questo corso, ma probabilmente in futuro, anche in minima parte, ne avremo molto probabilmente a che fare, quindi abbiamo trovato molto utile il seminario.

Marsiglia
Questo seminario ha avuto un focus molto più preciso sulle metodologie e la storia dell'ingegneria del software. E' stato molto interessante. Si è partiti analizzando i metodi più tradizionali, quindi waterfall e simili, per passare poi all'Agile e al DevOps, e quindi alla continuous delivery. Ha mostrato come cambiano queste varie metodologie, e come si adattano in base alla tipologia di ambiente in cui ci si trova. La metodologia va infatti adattata in base alla geografia, al team, alle tecnologie e ai progetti, come mostrato nella presentazione. Per questo esiste una metodologia ottimale per ogni situazione, e non è mai la stessa. Abbiamo potuto riscontrare questo anche negli altri seminari messi insieme. Abbiamo poi avuto un excursus sulle tecnologie cloud, di perchè e come sia rivoluzionario, e di come stia di fatto rendendo il DevOps sempre più preponderante come metodologia di sviluppo software. Questo perché come tipologia di progetto si adatta bene alle veloci messe in produzione, alla fase di testing, e continuous integration. E' stato un seminario interessante per entrambe le branche trattate.

APSS e Trentino.ai
In questo seminario è stato enfatizzato l'importanza del dipartimento tecnologie all'interno di un'azienda e di come questo influenzi l'organizzazione di quest'ultima.
Lo sviluppo di un software interno non si limita più all'analisi funzionale ma definisce il processo organizzativo dell'azienda, portando ad un co-design completo tra aspetto tecnologico e aspetto funzionale, ciò evidenzia l'importanza dell'organizzazione durante lo sviluppo di un software per ottimizzarne l'efficienza, favorire la collaborazione tra i diversi reparti aziendali, massimizzare l'utilizzo delle risorse disponibili e garantire una corretta implementazione delle best practices, contribuendo così al raggiungimento degli obiettivi aziendali complessivi e alla creazione di soluzioni software che soddisfano appieno le esigenze specifiche di ogni settore interno. Un altro tema importante è stato la sicurezza dei dati e della loro categorizzazione, concetto molto importante quando si sviluppa un software che deve conservare dati in maniera conforme alle politiche in vigore. L'ultima parte del seminario era incentrata sulle intelligenze artificiali, discutendo il grande contributo che porta al settore sanitario, possiamo intuire in realtà come queste possibilità non siano limitate a pochi settori ma possano trovare applicazione in ogni realtà aziendale. E' stato un seminario interessante, ovviamente gli argomenti sono stati trattati in maniera stretta al settore sanitario ma tutte le nozioni si possono applicare nei progetti di tutti i giorni e in ogni altro settore.

Il nostro approccio, in relazione a ciò che abbiamo imparato
Durante il corso del progetto e dei vari seminari, abbiamo imparato che i documenti, gli artefatti e la documentazione sono necessari per la buona uscita di ogni progetto. La comunicazione e l'essere sincronizzati sono chiavi per evitare disastri e perdite di tempo insostenibili. Va anche specificato che, però, non esiste una quantità perfetta di tutti questi elementi, in quanto variano molto a seconda del team, del progetto e del cliente finale. Per questo ogni caso ha la sua miscela ottimale, e si cerca sempre di migliorarla.

Ed è quello che abbiamo provato a fare anche noi. In particolare, abbiamo posto molto focus sulla sincronizzazione durante il lavoro, con feedback molto frequenti e revisioni continue del lavoro altrui. Non abbiamo prodotto documenti extra, oltre a quelli necessari al progetto, per evitare di burocratizzare eccessivamente il processo, e siamo rimasti molto agili. Direi che alla fine, il nostro approccio è stato una via di mezzo tra quello presentato in BlueTensor e in Meta. Forse il fatto che siano stati tra i primi ha giocato un ruolo in questo. Tutto sommato però, la nostra metodologia si è evoluta molto, con vari cambiamenti a mano a mano che il tempo passava, quindi a lungo termine, e a mano a mano che ci si evolve, è necessario anche avere la flessibilità di poter cambiare ed adattarsi. Secondo noi, questa è la chiave per avere un buon successo quando si parla di ingegneria del software.

Organizzazione del lavoro

Per quanto riguarda la pianificazione del lavoro abbiamo deciso di essere abbastanza elastici. In particolare, non solo abbiamo cercato di venire incontro alle esigenze extra universitarie di ognuno di noi, ma si è anche data la possibilità di lavorare maggiormente nelle parti del progetto in cui il singolo risultava più competente e interessato.

Si è seguito l'approccio per cui all'aggiunta di nuovi argomenti del corso, venivano aggiustate adeguatamente anche le sezioni dei deliverables.

Per quanto riguarda la spartizione del lavoro, si è deciso di fare uso di Asana, un software utilizzato a livello lavorativo per la fase di task management. Il lavoro veniva quindi suddiviso in dei piccoli task visibili su una dashboard da tutti i membri del gruppo. Ogni volta che un membro iniziava una nuova sezione, lo comunica via chat agli altri membri del gruppo.
Per essere più concreti, portiamo un esempio:
Example: se il task consisteva nello sviluppare il diagramma delle classi, veniva aggiunto alla dashboard di Asana da un membro del gruppo, si avvertiva il resto del team dell'inizio dello svolgimento della task in modo da evitare lavoro duplicato, e mano a mano il progetto si evolve.

Parlando invece dello sviluppo del codice, si è deciso di spartire il lavoro tra frontend e backend. La motivazione principale consiste nel fatto che due membri del gruppo presentavano maggiore manualità nello sviluppo della UI e integrazione con il backend rispetto al terzo, che aveva invece più esperienza nello sviluppo di REST API.

Le attività lavorative si sono svolte principalmente in modalità individuale.
Molte delle attività teoriche svolte non richiedevano una collaborazione attiva, ma solamente di rimanere coordinati nella creazione di ogni singola componente e sulla revisione finale.
Gli aggiornamenti all'interno del gruppo erano molto frequenti e avvenivano sia in modalità offline che online, via Asana e gruppo Telegram. Questo ha fatto sì che il lavoro fosse sempre sincronizzato e aggiornato.

Allo scadere della consegna di ogni documento ci impegnavamo a revisionare l'intero lavoro svolto in modo da assicurarci la coerenza non solo dei singoli deliverable ma soprattutto dell'intero progetto.

Ci siamo sempre organizzati in modo da avere come minimo uno studente che fosse presente a lezione. Le lezioni erano frequentate sia dal vivo che in remoto e molto spesso da tutti i membri del gruppo. Coglievamo a quel punto la possibilità di aggiornarci e di verificare la disponibilità della singola persona per future lezioni.

Per concludere questa sezione elenchiamo i software utilizzati nella fase di stesura dei deliverable, progettazione e sviluppo software:

- Google docs, abbiamo adottato questo strumento perché permette una veloce e intuitiva sincronizzazione e aggiornamento del lavoro insieme alla possibilità di inserire commenti ai membri del gruppo.
- Asana, questo tool è stato utilizzato principalmente come strumento di task management e per segnare le ore lavorative del singolo membro.
- Git/GitHub, ci hanno aiutato ad instaurare un ottimo workflow grazie alla funzionalità dei commit e al processo di CI/CD. Grazie a questi due tool si gestiva la parte relativa al codice e la memorizzazione dei deliverable.
- Figma, come strumento di progettazione del frontend dell'applicazione si è deciso di fare uso di questo potentissimo strumento per web design.
- LucidChart e FlowMap, ci siamo serviti di questi software per la realizzazione dei diversi diagrammi richiesti nei deliverable.
- Telegram, da non dimenticare questo strumento chat che ci ha permesso di comunicare efficacemente tra noi e di mantenerci aggiornati.

Ruoli ed attività

Riportiamo a questo punto i ruoli che ognuno di noi ha assunto durante lo sviluppo di tale progetto. Come detto precedentemente, per lo sviluppo la divisione dei compiti è molto marcata, per i vari documenti è stata molto più labile e flessibile, quindi di base ognuno ha avuto modo di dare un qualche contributo ad ogni parte, grande o piccolo che sia.

Componente del team	**Ruolo**	**Principale attività**
Ilya Emeliyanov	Project leader, analista requisiti funzionali e non, sviluppatore frontend	1. Si è occupato di coordinare il lavoro del gruppo, marcando lacune, inadeguatezze e incongruenze nei singoli componenti. Ha lavorato principalmente sul D1, D2 e D5. 2. Ha contribuito a stendere e sistemare le sezioni testuali dei singoli documenti. 3. Ha provveduto a realizzare buona parte del frontend. 4. Si è occupato della progettazione dei design mockup con Figma.
Ivano Lu	Analista requisiti funzionali e non, sviluppatore frontend	1. Si è occupato della progettazione dei design mockup con Figma. 2. Ha realizzato parte del frontend 3. Ha aiutato a stendere principalmente le sezioni dei documenti sul frontend
Matteo Possamai	Analista requisiti funzionali e non, sviluppatore backend	1. Ha contribuito principalmente al D3, D4 e al D5.

		2. Ha fornito la maggior parte dei diagrammi presenti in questi documenti, e in parte di quelli del D2. 3. Ha sviluppato il backend della piattaforma e la sua documentazione. 4. Creato la documentazione e il testing

Carico e distribuzione del lavoro

Riportiamo ora in tabella la distribuzione del lavoro per il progetto in maniera più deterministica, mediante le ore che abbiamo dedicato alla produzione dei vari documenti e del codice

	D1	**D2**	**D3**	**D4**	**D5**	**TOTALE**
Ilya Emeliyanov	19	23	24	43	3	112
Ivano Lu	20	23	20	36	3	102
Matteo Possamai	18	31	22	51	3	125
Totale	57	77	66	130	9	39

Criticità

Nel gruppo non sono state riscontrate notevoli criticità che abbiano rischiato di compromettere la realizzazione del progetto.
È giusto però annotare che ci sono solo state frizioni minori che mano a mano abbiamo appianato in modo da poter raggiungere il risultato migliore possibile.
Il problema iniziale che abbiamo dovuto affrontare consiste nel fatto che tutti i membri del gruppo lavorano su tutto e non vi era un efficace spartizione del lavoro. Questo, come si può intuire, ha portato ad un iniziale calo di produttività.
Perciò, in modo da ovviare al problema abbiamo deciso di dividerci i vari task a priori imponendo un forte focus su una sincronizzazione "asincrona" (via messaggi, asana, commenti google docs) ma comunque estremamente costante.

Questo ha permesso anche a chi avesse altri impegni personali di avere maggiore flessibilità e meno fiato sul collo.
Ognuno così si sentiva più libero e poteva decidere se lavorare più intensamente ad un deliverable ed eventualmente rallentare nella stesura di un altro.
Questo approccio ha permesso ad ognuno di noi di mettere mano in almeno una parte di ogni deliverable, anche se in maniera diversa.

Ilya ha partecipato maggiormente nella realizzazione dei documenti D1,D2 e D3, si è occupato di revisionare anche D4 e D5, ma in misura più ristretta.
Ivano ## TODO
Matteo si è concentrato prevalentemente nei D3, D4 e D5, e ha fornito anche buona parte dei diagrammi del D2. Ha scritto diversi paragrafi anche in D1 e D2, anche se in relazione alla sua contribuzione agli altri deliverable, qui risulta minore.

Per quanto riguarda lo sviluppo, la divisione marcata del lavoro ha permesso ad ognuno di noi di dare il massimo nell'ambito la cui expertise era maggiore.
In poche parole si è attuata una politica di "trust" per cui, chi si trovava a gestire il backend doveva solo preoccuparsi dello sviluppo del backend, dando per scontato il corretto funzionamento del frontend, e viceversa.
Ovviamente, prima dello sviluppo abbiamo concordato delle regole intra progettuali da seguire e le API comuni. Una criticità individuata è stata la necessità di modificare poi queste specifiche, ma in puro stile Agile. Siamo stati molto veloci ad aggiustare il tiro, e a permettere l'uscita del risultato migliore possibile.
Un altra piccola frizione è stata la poca conoscenza pregressa di alcune delle tecnologie proposte ed utilizzate, come ad esempio MongoDB. Nonostante ciò, la compartimentazione del lavoro, il feedback loop molto frequente e il costante lavoro, ci hanno permesso di superare questo limite.

Durante tutto il progetto, è stata mantenuta massima serietà e coordinazione. Abbiamo seguito le tempistiche e le scadenze per i singoli deliverable. Perciò siamo stati in grado di consegnare l'intero progetto per il primo appello in maniera efficace. Abbiamo tutti investito molte ore in queste attività e ne abbiamo imparato molto.

Siamo più che sicuri che queste conoscenze ci torneranno utili nel mondo del lavoro e in caso di successivi progetti di ingegneria del software saremo in grado di applicare le conoscenze acquisite in questo corso.

Autovalutazione

Data la mole di lavoro che ognuno ha sostenuto per mettere in piedi questo progetto, abbiamo fatto le seguenti considerazioni.

Abbiamo tutti lavorato un numero considerevole di ore, ed abbiamo avuto la possibilità di mettere mano a vari documenti. Ognuno ha avuto una sua contribuzione alla riuscita di tutto il progetto. Senza uno dei membri di questo team, questo progetto avrebbe avuto qualità peggiore. Inoltre, avendo provato a mettere le mani in molte cose, possiamo dire di avere approfondito tutti le nostre conoscenze relative all'ingegneria del software, sia dal punto di vista puramente teorico, ma anche più pratico, a livello di come funzionano i diagrammi e le documentazioni.
Abbiamo inoltre investito un considerevole ammontare di tempo per la parte di sviluppo dell'applicazione web che portiamo come prodotto di questo corso.
In conclusione, pensiamo di aver fatto tutti un buon lavoro. L'unica discriminante si può trovare nel numero di ore che ognuno ha dedicato. Nonostante ogni numero sia considerevole, alcuni lo sono più di altri, per cui ci è sembrato giusto fare una piccola differenza nei voti alla luce di questa considerazione.

Di conseguenza, la nostra autovalutazione è la seguente:

PERSONA	VOTO
Ilya Emeliyanov	30
Ivano Lu	30
Matteo Possamai	30