Progetto SDA

Si vuole sviluppare un sistema software in grado di automatizzare il processo di gestione della logistica dell'azienda al fine di migliorare la produttività e ridurre i tempi.

1. Presentazione del progetto

1.1 Processo attuale

Attualmente, ogni giorno, viene copiata una tabella dal software fornito da SDA e la stessa viene incollata su un foglio excel, dal quale abbiamo, inoltre, accesso a tutti gli ID e i dati dei vari giri. Successivamente vengono aggiornati, seguendo un template realizzato in precedenza, tutti i dati del giorno, quali: consegne, ritiri, ritiri speciali effettuati. Dopo aver effettuato tale aggiornamento il foglio esegue delle funzioni che ci permettono di calcolare: il fatturato del singolo giro, TM, MEDIA LV GG, CUS. Queste informazioni saranno utili a fini statistici e non.

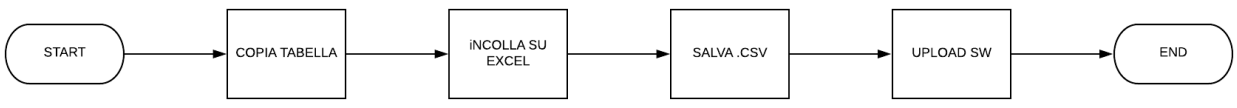
1.2 Problema

Un sistema del genere non ci consente di avere una certezza assoluta sulla validità dei dati, ne tantomeno sulla loro integrità, di conseguenza un sistema automatizzato ci permetterebbe di, intanto migliorare questi aspetti ma soprattutto di ridurre i tempi di gestione. Inoltre, il più delle volte i dati forniti dal software sda sono errati, quindi è quasi sempre necessario integrare gli stessi con una patch ricevuta via email.

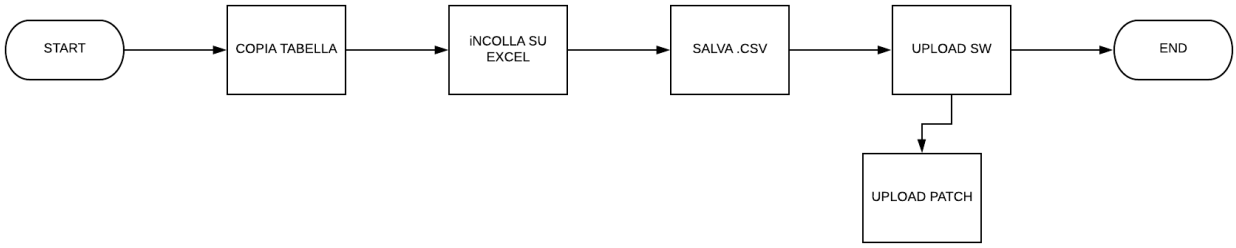
1.3 Possibile soluzione

Come possibile soluzione è stata pensata un'applicazione desktop in grado di poter automatizzare il tutto, in modo tale da preservare i dati e ridurre al minimo gli errori (circa 0,0001%). Iniziamo però vedendo l unico problema di fattibilità: purtroppo sda non ci consente di esportare la tabella dati visualizzata ogni giorno (infatti attualmente viene copiata a mano) quindi si dovrà pensare ad una soluzione ibrida. Una prima idea potrebbe essere quella di copiare e incollare la tabella su excel e poi salvarla in formato CSV, successivamente effettuare l'upload sul software fornito che a sua volta effettuerà tutti i calcoli del caso.

Lo schema dovrà essere il seguente:



Successivamente il software elaborerà i dati inseriti in input al fine di mostrare una tabella (seguendo il template già utilizzato attualmente). Aggiungerei personalmente un processo tra i 4 che prevede la possibilità, in caso di integrazione con una patch, di inserire un secondo file excel da confrontare con quello generato dalla copia della tabella di sda, cosi da mantenere alta la validità dei dati. Quindi:



Alla fine di questo processo avremo tutti i dati elaborati e tutti gli output richiesti.

2. Processo software

2.1 Descrizione del progetto

E' stata prevista la realizzazione di un'applicazione desktop, da installare tramite un eseguibile .exe
Sarà eseguibile in ambiente microsoft e sarà graficamente organizzata come concordato. Iniziamo elencando quelle che sono le principali funzionalità, tuttavia il software rispetterà tutte quelle che sono le caratteristiche tali da poter ottenere una manutenzione e degli aggiornamenti costanti e semplici nel tempo:

1. Upload file sda
2. Elaborazione file sda
3. Visualizzazione elaborazione file
4. Report export
5. Inserimento nuovo corriere (con tutte le sue variabili e attributi)
6. Confronto file sda e file patch
7. Modifica manuale delle tabelle elaborate
8. Login
9. Logout
10. Gestione profilo e account personale

2.2 Qualità del software

Oltre a seguire tutti quelli che sono i punti del ciclo di vita di un software (di cui parlerò nel prossimo punto) è importante capire quelle che sono le qualità del software in progettazione e sviluppo al fine di comprenderne la qualità. Il software avrà le seguenti caratteristiche:

- Modificabilita': il software avrà la possibilità di essere aggiornato molto semplicemente, il costo in tempo della progettazione sarà più alto del costo di sviluppo, tuttavia questo ci consentirà, tra le altre cose, di poter aggiungere moduli funzionali senza troppi sforzi.

- Conformità: Il software si adatterà all'ambiente esterno, all'evoluzione delle interfacce hardware o più utenti con profili diversi.

- Affidabilità: Un sistema è tanto più **affidabile** quanto più raramente, durante l'uso del sistema, si manifestano malfunzionamenti.

- Robustezza: In termini generali, la **robustezza** di un sistema è la misura in cui il sistema si comporta in modo *ragionevole* in situazioni impreviste, non contemplate dalle specifiche.

- Efficienza: Un sistema è **efficiente** se usa memoria, CPU e altre risorse in modo proporzionato ai servizi che svolge, ovvero senza sprechi. Il termine **prestazioni** ha un significato correlato più specifico; le prestazioni, infatti, sono da considerarsi come uno degli elementi che potrebbe essere specificato dai requisiti (si parla in questo caso di requisiti non funzionali).

2.3 Attività

Ci limitiamo qui ad elencare, suddividendole in tecniche e organizzative, tutte le attività previste durante lo sviluppo del progetto:

Attività tecniche:

– Analisi dei requisiti (requirements analysis)

– Progettazione (design)

– Realizzazione (implementation)

– Collaudo (testing)

– Messa in esercizio (deployment)

– Conduzione operativa (operation)

– Manutenzione (maintenance)

Attività organizzative:

– Gestione del progetto (project management)

– Gestione della configurazione (configuration management)

– Assicurazione della qualità (quality assurance)

2.4 Ambiente di sviluppo

L'ambiente di sviluppo previsto è l'IDE eclipse, che ci consente di sviluppare il software in ambiente Java (linguaggio di programmazione). L'architettura del software prevede diversi livelli, in particolare abbiamo il livello server e il livello DB. All'interno del livello server potrà essere necessario sostituire PHP a Java come linguaggio di programmazione per motivi di efficienza. Inoltre il livello DB sarà sviluppato in SQL.

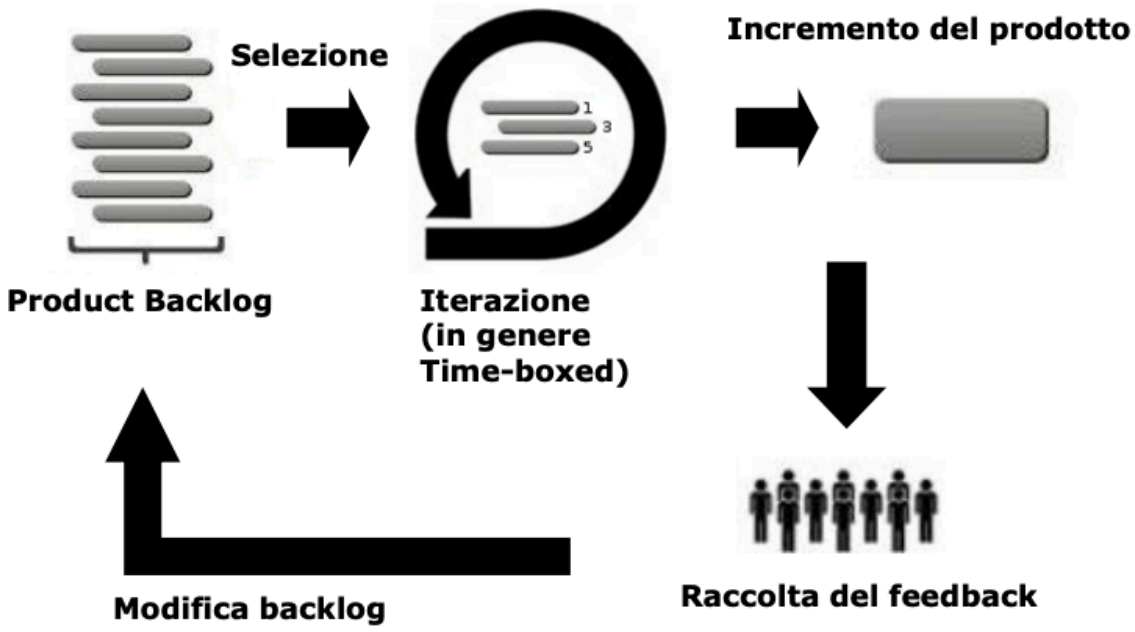
Nel caso in cui le fasi di analisi dei requisiti e progettazione ci rivelasserò la forte utilità nella comunicazione con microsoft excel, potremmo scegliere di sostituire l'ambiente Java con ambiente microsoft e C# come linguaggio di programmazione.

2.5 Sviluppo AGILE E SCRUM

La metodologia agile, nell'ambito dell'ingegneria del software, ci consente di sviluppare i nostri progetti nel modo più efficiente e veloce possibile, riducendo gli errori al minimo e rispettando il più possibile le esigenze del cliente.

La metodologia agile divide il processo software in più iterazioni, ogni iterazione fornisce un prodotto potenzialmente pubblicabile, questo ci consente di dividere il prodotto in rilasci, ognuno di questi funzionante, in modo tale da tracciare tutte le modifiche e rilasciare il prodotto (o comunque una parte) nel modo piu veloce possibile.

"Stiamo scoprendo modi migliori di creare software, sviluppandolo e aiutando gli altri a fare lo stesso. Grazie a questa attività siamo arrivati a considerare importanti: Gli individui e le interazioni più che i processi e gli strumenti Il software funzionante più che la documentazione esaustiva La collaborazione col cliente più che la negoziazione dei contratti Rispondere al cambiamento più che seguire un piano Ovvero, fermo restando il valore delle voci a destra, consideriamo più importanti le voci a sinistra."



Scrum è un framework agile per la gestione del ciclo di sviluppo del software, iterativo ed incrementale, concepito per gestire progetti e prodotti software o applicazioni di sviluppo. Scrum enfatizza tutti gli aspetti di gestione di progetto legati a contesti in cui è difficile pianificare in anticipo. Vengono utilizzati meccanismi propri di un "processo di controllo empirico", in cui cicli di feedback che ne costituiscono le tecniche di management fondamentali risultano in opposizione alla gestione basata sul concetto tradizionale di command-and-control. Il suo approccio alla pianificazione e gestione di progetti è quello di portare l'autorità decisionale al livello di proprietà e certezze operative.[2]

Il framework scrum sarà utilizzato nel nostro progetto al fine di implementare la metodologia agile. Sarà inoltre utilizzato GitHub come versionatore.

3. Ciclo di sviluppo e caratteristiche

3.1 Ciclo di vita del software

Il ciclo di vita del software racchiude tutte quelle che sono le fasi di analisi, di ricerca, di progettazione (con tutte le caratteristiche ad essa associate), sviluppo, test, rilascio, manutenzione e supporto.

3.2 Analisi dei requisiti

La fase di progettazione racchiude al suo interno delle importanti fasi, tra queste l'analisi dei requisiti. I requisiti si dividono il requisiti funzionali e requisiti non funzionali. I requisiti non funzionali sono quelli descritti nel punto 2.3, i requisiti funzionali invece sono tutte quelle caratteristiche che il software deve avere, quindi quelle descritte nel punto 2.1.

L'analisi dei requisiti è una fase in cui dovranno essere abbastanza frequenti, dei colloqui con il committente, questo perché è molto importante capire quelle che sono le esigenze, in modo tale da poter trovare una soluzione appropriata.

3.3 Design

La fase di design è la vera e propria fase di progettazione, la fase in cui viene progettato, attraverso diagrammi di vario tipo, il software, sia dal punto di vista concettuale, che in prospettiva software. E' una delle fasi più tecniche del progetto.

3.4 Implementazione

La fase di implementazione è la fase di sviluppo, la fase in cui, utilizzando l'ambiente descritto e il linguaggio di programmazione scelto, viene scritto il codice del programma.

3.5 Test

La fase di test è la fase più importante del progetto, del processo e del prodotto software. La fase di test non consiste solo nel testare il programma dal punto di vista

funzionale e non ha come obiettivo quello di non trovare errori, ma, al contrario, ha come obiettivo quello di trovare il maggior numero di errori possibile. E' pura utopia pensare ad un software senza errori. In questa fase potrebbe essere necessario coinvolgere utenti esterni.

3.6 Fasi intermedie

Vi sono una serie di fasi intermedie che si collocano tra le fasi descritte nei paragrafi precedenti, che possono essere utili durante lo sviluppo del software (es. incontri, videocall, raccolta di feedback o altre attività tecniche)

4. Manutenzione ed evoluzione

4.1 Tipi di manutenzione

La manutenzione del software può essere di diverso tipo.
La manutenzione del software è il processo di modifica di un prodotto software dopo il suo rilascio in esercizio.

• Impatto della manutenzione

– Ordinario

– Straordinario

Tipi di manutenzione

– Manutenzione correttiva (Eliminazione dei bug)

– Manutenzione adattiva (Adeguamento a cambiamenti nell'ambiente)

– Operativo: es. modifiche hw/sw di base (porting)

– Normativo: es. introduzione Euro, nuove leggi

– Manutenzione preventiva (Miglioramenti per ridurre il debito tecnco)

– Manutenzione perfettiva (Cambiamenti alle funzionalità richiesti dagli utenti per mantenere alta la soddisfazione)

4.2 Costi della manutenzione

Il supporto per la manutenzione ordinaria è gratuito. I costi degli altri tipi di manutenzione, che si riferiscono più che altro, all'evoluzione del software hanno un costo a progetto che dovrà essere successivamente discusso.

5. Tempi di sviluppo e costi

Sono stati previsti 6 mesi-persona. Suddivisi in vari step di rilascio.

Il primissimo rilascio è il seguente documento, dopodichè sarà rilasciato il prototipo del programma e quindi 5 step tra progettazione, sviluppo e test.