Corso di Laurea in Ingegneria dell'Informazione

Relazione del tirocinio presso AVL

Laureando:

Silvia Dal Molin

Relatore:

Professor Massimo Rumor

Anno Accademico 2012/2013

Università degli Studi di Padova

Dipartimento di Ingegneria dell'Informazione

Corso di Laurea in Ingegneria dell'Informazione

Relazione del tirocinio presso AVL

Laureando:

Silvia Dal Molin

Relatore:

Professor Massimo Rumor

Anno Accademico 2012/2013

Indice

Introduzione **1**

1 MKS **3**
1 MKS concetti chiave 4
1.1 Item 4
1.2 Workflow 4
1.3 Sicurezza 5
1.4 Modelli e simulazioni 5

2 Obiettivi del Tirocinio **7**
1 Gestire l'accesso a uno solo dei quattro server di MKS 7
2 Gestione delle licenze di accesso 9
3 Permessi standard per tutti i progetti 9
4 Statistiche sull'utilizzo delle licenze 10
5 Creare nuovi progetti 10

3 Tecnologia Utilizzata **13**
1 Python 13
2 PyGTK 14
3 Comunicare con MKS 15

4 Lavoro svolto **17**
1 Creare un nuovo progetto 17
1.1 La richiesta 18
1.2 Cos'é il MAPSEC? 19
1.3 La non modificabilitá dell'Integrity Server 20
1.4 Conferma creazione 20
2 Errori dovuti ai Permessi 21
2.1 Controllo dei permessi 22
2.2 Uso del Log File 22

2.3 Leggere il problema 23

Conclusioni **25**

Bibliografia **27**

Elenco delle figure

1.1 Esempio di un item 5
1.2 Workflor standard di un progetto 6

2.1 Casi di accesso 8

3.1 Codice python per la creazione di una finestra 15
3.2 La finestra creata processando il codice 15
3.3 Codice Python per la chiamata del terminale 15

4.1 Schema per la creazione di un nuovo progetto 18
4.2 Finestra per la creazione di un nuovo progetto 19
4.3 Schema per l'individuazione e soluzione di un problema dovuto ai permessi 21

Introduzione

Dal Marzo 2011 all'Agosto 2012 ho svolto un tirocinio presso la ditta AVL. AVL é una multinazionale con sede centrale a Graz in Austria, i suo prodotti vanno dal power engineering, all'automazione, all'ingegneria per l'automobile. Sebbene la sede centrale sia in Graz, il team informatico ha come riferimento la sede di Ratisbona, dove ho svolto il mio tirocinio. Il team a cui appartenevo, chiamato MKS team, ha come compito la gestione e mantenimento del software e, inoltre, gestisce la formazione degli utenti MKS.

Essendo AVL un multinazionale, i team di sviluppo spesso non sono fisicamente dislocati nello stesso stato. Questo genera tutta una serie di problematiche, sopratutto di comunicazione, infatti bisogna essere sicuri che si sta lavorando in parallelo e che tutti i componenti del team stiano lavorando sulla stessa cosa. MKS ha proprio questo compito: permettere a tutti e soli i membri del team di vedere, controllare, modificare tutte le componenti del loro progetto, dalla richiesta del cliente, proseguendo per lo sviluppo, testaggio e, infine, controllo qualitá e brevetti. MKS ha una struttura molto complessa e articolata, per questo il primo approccio al software é molto traumatico per un nuovo utente, perció non tutti i team di AVL hanno accesso a MKS, ma solo alcuni settori, con il progetto di espanderne l'utilizzo a tutta l'azienda.
Mi é stato chiesto di svolgere, durante la mia attivitá di tirocinio, una duplice funzione: da un lato il mio compito é stato quello di scrivere alcuni codici per la creazione di alcune GUI, quindi sono stata un semplice "developer"dall'altro mi é stato chiesto di seguire gli utenti MKS alle prime armi.

In questa relazione voglio illustrare, in primo luogo, la parte piú tecnica di quella che é stata la mia esperienza, quindi la presentazione del linguaggio Python e in particolare le interfacce grafiche (GUI) da me create, in particolare cosa mi é stato richiesto: problemi-soluzioni. In una seconda parte, invece, la gestione utenti: ottenere le licenze per l'accesso al sistema, avere i permessi per vedere, modificare, approvare un progetto salvato nel server e richieste di apertura di nuovi progetti all'interno del sistema.

Capitolo 1

MKS

MKS é un prodotto di PTC, ma fu inizialmente sviluppata da MKS Inc, da cui prende il nome, la quale fu acquisita successivamente da PTC. MKS é un sistema software per la gestione del ciclo di vita e Application Lifecycle Management (ALM). Il software é client / server, sia con desktop (java / swing), sia con interfacce client web. Esso offre alle aziende un ambiente collaborativo in cui possono gestire i processi end-to-end di sviluppo, dalla gestione dei requisiti, gestione del cambiamento, revisione,la costruzione e la gestione del testaggio e la distribuzione del software, nonché le statistiche di produttivitá sul lavoro svolto e i documenti finali di un progetto. MKS é una delle aziende che per prime hanno sviluppato un software per la gestione del ALM (*Application Lifecyle Managment*), la prima versione risale al 2001.

Ad ogni Project Manager, MKS, fornisce molti format per contenere quelli che secondo MKS sono i dati base per lo sviluppo di un qualsiasi progetto.
Questi sono:

- Requisiti e obiettivi del progetto
- Codice, modelli, ecc...
- Test e risultati
- Documentazione
- Informazioni sul Project Management

Per aumentare le prestazioni e la salvaguardia dei dati inseriti, il software si struttura tra due diversi server, tra loro fortemente connessi.
Il primo, chiamato MKS Integrity, che contiene lo sviluppo vero e proprio del progetto. Qui il project manager puó gestire il lavoro del suo team creando

obbiettivi e affidando, di conseguenza, i vari compiti ai componenti del team, potendo cosí stabilire un andamento temporale del progetto.
Il secondo, chiamato MKS Source, questo server é stato concepito piú per il lavoro quotidiano degli sviluppatori, permettendo cosí veloci accessi, cambiamenti del prodotto, sincronizzazioni, ecc....

Un altro forte aiuto, che questo software da, é la possibilitá di accedere a tre diversi tipi di server:

1. mks training
2. mks test
3. mks source

Come si puó facilmente dedurre dal nome, questi tre server servono per lo sviluppo di diverse fasi di un progetto.

1 MKS concetti chiave

1.1 Item

Un Item é l'elemento fondamentale di MKS, infatti in questa struttura ci sono tutti gli attributi del progetto:informazioni di gestione, utente assegnato, stato, date di completamento, scopo, descrizione, classificazione, ecc.. e collega fra di loro dati, codici, progetti di sviluppo, ecc.. Ogni item ha un tipo assegnato; esempi includono Requirement, Specification, Test Case, richiesta di modifica, ecc.. Gli Item sono molto utili in quanto sono collegambili tra di loro e quindi si riesce, tramite essi, a costruire la struttura di un progetto.

Le relazioni tra Item possono creare anche una gerarchia all'interno di un progetto, cioé si usa un item come contenitore principale e poi usare degli altri item per dividere il progetto in parti piú piccole. Esiste anche la possibilitá di raggruppare un intero set per obiettivi sotto un unico item, questo aiuta l'analisi di gestione del progetto.

L'ultimo interessante aspetto degli item é la possibilitá di essere riutilizzati all'interno di piú progetti o documenti. Questo genera automaticamente una gerarchia padre-figlio tra i progetti.

1.2 Workflow

Ogni Item ha un proprio workflow, che descrive come si muove un progetto da stato a stato. Per avere una transizione da uno stato ad un altro é necessario

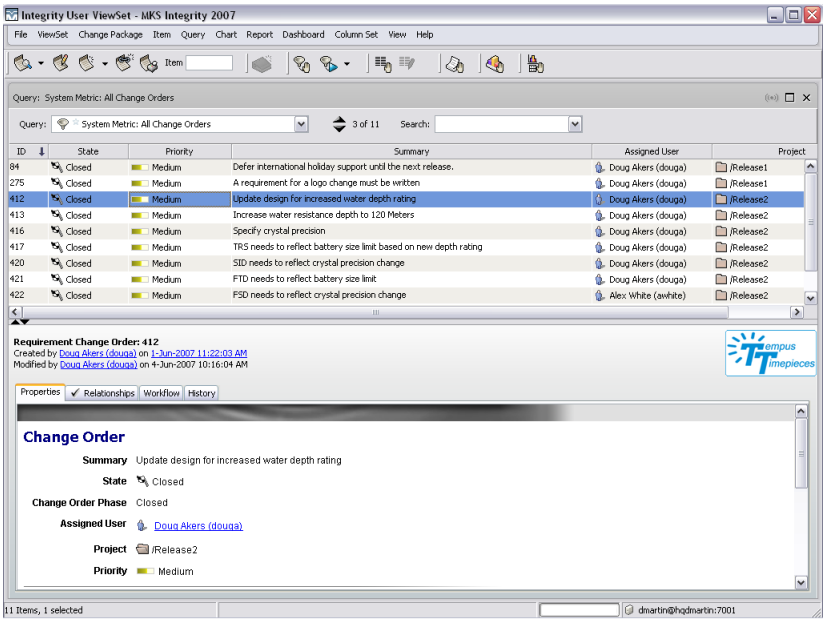

Figura 1.1: Esempio di un item

rispettare regole precise; le condizioni e le autorizzazioni necessarie sono definite dall'amministratore e applicate dal sistema. Il workflow descrive la storia del'item e viene visualizzato agli utenti, quindi si puó capire a che punto é lo sviluppo dell'item e durante le verifiche periodiche si possono analizzare i problemi e come sono stati risolti.

1.3 Sicurezza

Ogni progetto e Item ha un determinato livello di sicurezza, questo vuol dire che si possono attribuire dei diversi livelli di accessibilitá a un progetto. Questo permette ad ogni azienda di poter stabile un livello standard di pivacy per i suoi progetti, mantenendo comunque la possibilitá di cambiare le condizioni di sicurezza di un determinato progetto. Questo consente di mantenere degli alti standard di privacy.

1.4 Modelli e simulazioni

MKS, grazie alla sua struttura, da la possibilitá di eseguire dei testaggi direttamente, ció significa che dalla finestra principale si possono lanciare programmi come Matlab e Simulink per l'esecuzione dei codici eseguiti. Questo avvantaggia l'utente in quanto vengo automaticamente registrati input e output del

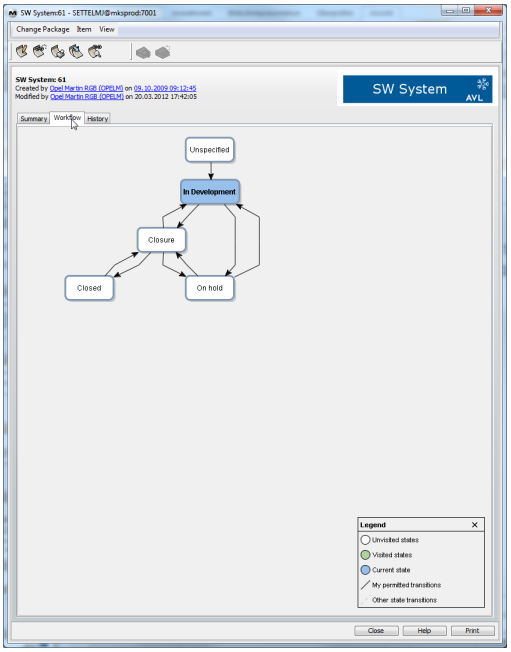

Figura 1.2: Workflor standard di un progetto

testaggio, potento cosí facilmente analizzare dove ci sono delle lacune o eventuali miglioramenti possibili. L'aspetto piú interessante dell'integrazione con Matlab e Simulink risulta, oltre al vero e proprio lancio del programma, sta nel poter tranquillamente aprire i file creati con questi programmi anche con MKS. Questa interazione, molto utile per gli sviluppatori, permette, ad esempio, di risparmiare tempo per un utente che deve solo controllare i codici/progetti e non appesantisce le prestazioni del proprio pc, evitando l'esecuzione di troppi programmi contemporaneamente.

Capitolo 2

Obiettivi del Tirocinio

Soraja é il nome che é stato dato alla prima GUI da me scritta. Questa interfaccia ha il compito di supportare l'interfaccia Admin, fornita direttamente dal software MKS, infatti quest'ultima presenta una serie di lacune che rendono l'interfaccia difficile da usare. In questo capitolo verranno spiegate le principali funzionalitá di Soraja.

1 Gestire l'accesso a uno solo dei quattro server di MKS

In MKS esiste la possibilitá di collegarsi a piú server contemporaneamente, questo crea un problema in quanto un utente distratto, spesso non sa su che server sta lavorando.
Questo crea dei disagi, ad esempio se un utente vuole modificare il codice che é in testaggio ed é connesso sia al Source che al Test, essendo il server Source il default server, se l'utente non cambia eplicitamente la connessione, modificherá il codice presente nel server Source.
Mi é stato chiesto, perció, di programmare per la GUI una classica finestra per il login, ma con accesso mirato ad un unico server, cioé di rendere impossibile la connessione a due o piú server contemporaneamente, e se l'utente aveva giá fatto un login tramite la GUI rendere possibile il recupero della password senza digitarla nuovamente.

Per quanto riguarda la gestione del login ai server, questa finestra deve poter gestire tre diverse situazioni base:

1. l'utente non é connesso a MKS, quindi bisogna fare il login;

2. l'utente é connesso ad MKS, ma é connesso a piú server, quindi bisogna disconnettersi da tutti i server, tranne uno, e ricordare in quale server si sta lavorando;

3. l'utente é connesso ad MKS ed é connesso ad un solo server, chiedere all'utente se vuole lavorare su quel determinato server o se vuole cambiare server.

La *Figura* 2.1 illustra tutte le possibili situazioni gestite dal login.

Connection Case Table

COMPUTER LOGIN	DATASTORE		CONNECTION		
USER	**USER**	**PASSWORD**	**There is connection+ Asked user**	**USER**	**PASSWORD**
A	A	PA	True-A	A	PA
A	A	PA	False-A	A	PA
A	A	-	True-A	A	-
A	A	-	False-A	A	-
A	-	-	True-A	A	-
A	-	-	False-A	A (ask comp)	-
A	B	PB	True-A	A	-
A	B	PB	False-A	A (ask comp)	-
A	A	PA	True-B	B	-

Figura 2.1: Casi di accesso

Dove *Data Store* é un file in cui vengono salvati tutti i dati (criptati) dell'utente, cio é usernale, password, ecc... Questo file viene creato appositamente per facilitare un successivo login dell'utente.

2 Gestione delle licenze di accesso

Tutte le informazioni e tutti i progetti salvati tramite MKS vengono immagazzinati in un unico server centrale. Ovviamente l'azienda MKS non fornisce questo servizio solo ad AVL, ma a molte altre aziende. Per regolare gli accessi al server centrale MKS "vende"delle licenze di accesso, quindi se un utente vuole accedere ai dati salvati o vuole salvare dati deve avere una di queste licenze. Il problema che si poneva alla AVL era che c'era 25 licenze per oltre 400 utenti MKS, quindi nelle ore di punta era frequente che un utente non potesse accedere al sistema. Uno strumento fornito agli Admin, direttamente da MKS, é quello di poter vedere gli utenti che hanno MKS aperto, chi sta usando attivamente una licenza e chi sta usando una licenza, ma non sta facendo nulla.
Dall'analisi di questi dati, si é riuscito a dedurre che se un utente entrava e quindi riceveva una licenza, ne rimaneva in possesso per 30 minuti, anche se inattivo, e poi veniva disconnesso automaticamente dal software. Il problema sussisteva che agli admin non era possibile modificare i 30 minuti per la disconnessione.
Per risolvere ció, é stato creato un algoritmo per cui ogni 30 secondi vengono controllati tutti gli utenti connessi a MKS e chi risulta inattivo per piú di 10 minuti, viene disconnesso automaticamente. Questo ha reso le 25 licenze piú che sufficienti per tutti i 400 utenti.

3 Permessi standard per tutti i progetti

All'interno di MKS vengono salvati tutti i progetti in sviluppo in AVL, ovviamente come una qualsiasi azienda ci sono delle severe e strette politiche della privacy. Per ottenereil controllo della privacy ad ogni progetto vengono assegnati dei gruppi, solo chi é all'interno di questi puó vedere, accedere o modificare un progetto.
I gruppi di ogni progetto sono:

- Team Leader (ne esiste uno per ogni progetto) é il coordinatore ufficiale del Team;
- Team (ne esiste uno per ogni progetto) é il gruppo contenente gli sviluppatori del progetto;
- Admin sono gli Amministratori di MKS;
- Team Quality é un team, i cui componenti hanno il compito di controllare la qualitá del prodotto.

Ovviamente ognuno dei quattro gruppi ha dei permessi diversi.
Per esempio: non ha senso che un membro del team quality modifichi un codice di un progetto, ma sta a lui modificare il report sulla qualitá, quindi gli saranno negati dei permessi riguardo la modifica del progetto e avrá dei permessi per modificare gli item, tra cui il report finale del progetto.
L'assegnazione dei permessi é esclusivo compito degli admin, questo creava notevoli disagi in quanto con un elevato numero di progetti era alquanto difficile gestire i permessi di tutti. Si é deciso quindi di creare uno strumento che permetta di scegliere quale autorizzazione applicare a ciascuno dei quattro gruppi e applicarli automaticamente a tutti i progetti scelti.

4 Statistiche sull'utilizzo delle licenze

Come precedentemente detto il problema di avere 25 licenze per oltre 400 utenti creava notevoli disagi. Il problema, per la sede centrale, sussisteva nel capire se effettivamente tutti gli utenti registrati usassero realmente il programma, in che modo, con che frequenza e, sopratutto, quale fosse la filiale che usava maggiormente il programma, per poter così decidere dove meglio dislocare il team MKS. Mi é stato chiesto, quindi, di stimare le ore di utilizzo di MKS per ogni utente e filiale. Nella finestra creata é possibile selezionare un periodo di tempo d'interesse, attraverso la scelta di una data di inizio e fine. Vengono, cosí, generate due tabelle.
La prima tabella contiene tutte le ore di utilizzo di MKS divise per utente, mentre la seconda contiene le ore per filiale.
In questa finestra si puó ricavare, anche una serie di grafici, i quali stimano l'utilizzo medio di ogni tipo di licenza nell'arco temporale selezionato. Grazie all'analisi di questi grafici é possibile capire quali licenze sono piú usate e quali meno, al fine di aumentare o ridurre il numero di un determinato tipo di licenze.

5 Creare nuovi progetti

Essendo MKS molto strutturato come programma, una semplice richiesta di aprirtura di un nuovo progetto, compito esclusivo degli admin, si doveva:

1. Controllare l'esistenza dei gruppi, la cui creazione é compito del team leader

2. Creare una nuova cartella, items e progetti per entrambi i server (Integrity e Source)

3. Applicare i permessi

Questi tre semplici passi risultano molto laboriosi e ripetitivi, se considerati su larga scala. Si é creata, perció, la possibilitá di creare automaticamente un nuovo progetto. Ricalcando, nell'interfaccia, la struttura del modulo che un utente deve compilare per richiedere un nuovo progetto.

Le varie parti di questa interfaccia verranno spiegate e approfondite nel *Capitolo 4*.

Capitolo 3

Tecnologia Utilizzata

1 Python

Python é un linguaggio di programmazione fortemente orientato ad oggetti (OOP), in cui vengono bene supportati i concetti di polimorfismo, overloading degli operatori ed ereditarietá multipla. Tutte queste caratteristiche, piú la sua semplice e chiara sintassi, rendono l'implementazione dell'OOP banale e intuitiva da applicare.
Un'altra particolaritá é l'essere un'Open Source, questo, a primo acchito, potrebbe risultare un fatto negativo, ma non lo é. Infatti, sebbene sia libero, é molto bene supportato dalla corposa comunitá che si é sviluppata in questi anni. L'utilizzo di Internet, come mezzo principale di comunicazione di questa comunitá, rende facile e, quasi, immediato trovare risposte a dubbi o soluzioni a problemi. Un problema che si potrebbe porre é l'incontrollata modifica dell'implementazione del linguaggio. Questo non avviene in quanto, per poter cambiare, il linguaggio bisogna seguire una formale procedura chiamata PEP, per poi essere analizzata e approvata dal BDFL (Benevoled Dictator For Life, che é l'ideatore: Guido van Rossum), tutto ció rende il Python restio ai cambiamenti.

Spesso la scelta di questo linguaggio, rispetto ai moltissimi disponibili, viene fatta perché tutti gli programmi scritti in Python si compilano e girano in tutte le principali piattaforme in uso, ció e possibile perché l'implementazione standard é scritta in ASCI C portabile.

Dal punto di vista delle sue funzionalitá, il Python si trova a metá tra i linguaggi di scripting e i linguaggi di sviluppo di sistemi. Gli strumenti principali messi a disposizione dal linguaggio sono:

- Tipizzazione dinamica: Non si dichiara una variabile e neppure il tipo;

- Garbage collection: elimina dalla memoria gli oggetti non piú usati;
- Classi, moduli ed eccezioni liberi da usare;
- Tipologie di oggetti precostruite: liste, dizionari e stringhe flessibili e facili da usare;
- Strumenti precostruiti: l'unione di collezioni, l'estensione di sottoparti, l'ordinamento, la mappatura, ecc...
- Ampie librerie;
- É facilmente mischiabile: posso inserire componenti scritti con altri linguaggi.

Ovviamente esistono anche degli svantaggi nell'usare il linguaggio Python. Il principale é sicuramente la velocitá di esecuzione dei programmi. Rispetto al C o al C++, l'esecuzione dei programmi Python risulta certamente piú lenta, in quanto la compilazione prevede la traduzione del codice sorgente in un formato intermedio, detto byte code, per poi essere interpretato. L'esistenza del byte code rende possibile la portabilitá nelle diverse piattaforme, ma siccome non é compilato fino a livello di codice macchina binario i programmi verranno eseguiti piú lentamente.
Tuttavia il Python viene eseguito in modo sufficientemente velocemente per la maggior parte dei programmi.

2 PyGTK

Il mio team ha scelto di usare il Python, come linguaggio di programmazione per tutti i motivi citati nel paragrafo precedente, ma sopratutto perché lo sviluppo di GUI (Graphical User Interface) é supportato da appositi moduli, inoltre le GUI create non mutano il loro aspetto quando vengono lanciate da piattaforme diverse. In particolare ho usato molto il pacchetto PyGTK.

Questo pacchetto é un insieme di moduli Python per l'implementazione di interfacce in GTK+. GTK (GIMP Toolkit) é una libreria creata appositamente per la creazione di interfacce grafiche. il nome, GIMP Toolkit, deriva dal fatto che é stato scritto in origine per sviluppare il GNU Image Manipulation Program (GIMP), tuttavia GTK ora é utilizzato in un gran numero di progetti di software, tra cui la GNU Network Object Model Environment (GNOME).

GTK é sostanzialmente un API (application programmers interface), scritta in linguaggio C esteso e implementata seguendo il paradigma della programmazione orientata ad oggetti.

```python
import gtk

class Base:
    def __init__(self):
        self.window = gtk.GtkWindow(gtk.WINDOW_TOPLEVEL)
        self.window.show()

def main():
    gtk.mainloop()

print __name__
if __name__ == "__main__":
    base = Base()
    main()
```

Figura 3.1: Codice python per la creazione di una finestra

Figura 3.2: La finestra creata processando il codice

```python
def sh(self, cmdline):
    """
        Run a given command in a subprocess and return its out_message.
        raises RuntimeError on error.
    """
    def _kill_process_after_a_timeout():
        subprocess.Popen("taskkill /F /T /PID %i" % p.pid, shell=True)
        self.timeout = "Timeout"
        return

    self.timeout = ""

    p = subprocess.Popen(cmdline, shell=True, stdin=subprocess.PIPE ,stdout=subprocess.PIPE,
                         stderr=subprocess.PIPE,universal_newlines=True)

    watch = threading.Timer(120.0, _kill_process_after_a_timeout)
    watch.start()
    stdout, stderr = p.communicate()

    watch.cancel()

    if self.timeout == "Timeout":
        raise TimeoutError('\nTimeout: ' + str(cmdline))

    if p.returncode != 0:
        raise RuntimeError(stderr)

    error_message = str(stderr) + '\ncmd line: ' + str(cmdline) + '\n'

    return  stdout, error_message
pass
```

Figura 3.3: Codice Python per la chiamata del terminale

Nella *Figura* 3.1 possiamo vedere come in poche righe di codice si riesca a creare una finestra perfettamente funzionante, *Figura* 3.2, grazie all'esistenza di un'apposita classe (GtKWindow) ha un unico scopo: la gestione di finestre.

3 Comunicare con MKS

Oltre alla parte grafica si é creato un metodo che permetta di interagire con il terminale del pc, il quale grazie ad apposite righe di comando, riesce a comunicare o richiedere dati a MKS.

Il metodo riportato in *Figura* 3.3 deve avere come argomento la riga di comando, che si desidera lanciare, sotto forma di stringa e si riceve come output due stringhe:
la prima contenente il vero e proprio output, sotto forma di stringa, con cui ci interessa lavorare,
il secondo una stringa contenente degli errori secondari, cioé errori non fatali per il programma.

Capitolo 4

Lavoro svolto

1 Creare un nuovo progetto

Come spiegato nel *Paragrafo 2.5* in Soraja é presente una classe che, oltre a gestire un'apposita finestra, fa si che venga creato un nuovo progetto in MKS. Prima della creazione di questo strumento, per ottenere lo stesso risultato un Admin doveva:

1. Verificare la correttezza della richiesta arrivata dall'utente
2. Verificare l'esistenza dei gruppi attraverso una ricerca nell'elenco completo di tutti i gruppi
3. Creare una cartella nel server MKS-Source
4. Creare una cartella nel server MKS-Integrity
5. Creare gli Item che colleghino i progetti del source e dell'integrity, contenenti gli obiettivi e i nomi degli sviluppatori del progetto
6. Settare, manualmente, i permessi
7. Dare il feedback all'utente.

Per fare tutto ció era necessaria piú di un'ora, con lo strumento presente in Soraja, invece, riportando correttamente le informazioni contenute nel modulo di richiesta, é possibile ottenere lo stesso risultato molto piú velocemente.

Tutte le operazioni descritte nell'elenco precedente, sono eseguite in automatico dal programma.
La *Figura* 4.1 illustra in un semplice schema tutti gli step necessari per creare un nuovo progetto.

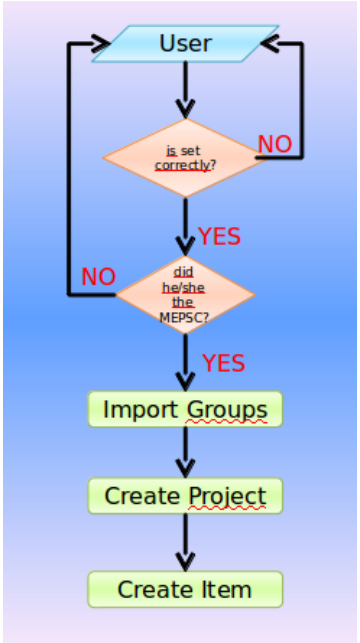

Figura 4.1: Schema per la creazione di un nuovo progetto

1.1 La richiesta

Come si vede in *Figura* 4.2 i campi che un utente deve fornire per l'apertura di un progetto non sono molte, tuttavia spesso il modulo di richiesta arriva incompleto o errato. Le informazioni assolutamente necessarie per la creazione sono:

- Un codice numerico che identifica il cliente
- Un codice alpha-numerico che identifica il progetto
- Uno short name, cioé poche parole che identificano brevemente il progetto
- La necessitá o meno di creare un progetto parallelo per i Project Manager
- La necessitá o meno di usare progetti giá esistenti

La compilazione della richiesta é un'operazione che sembra apparentemente banale, ma spiegare ad un utente come farlo correttamente in che modo, era ed é uno dei compiti piú complessi e dispendiosi in termini di tempo. I principali problemi riguardavano il nome dato al progetto. I progetti contenuti in MKS, come prima accennato, sono distinti da un codice numerico, che indica il

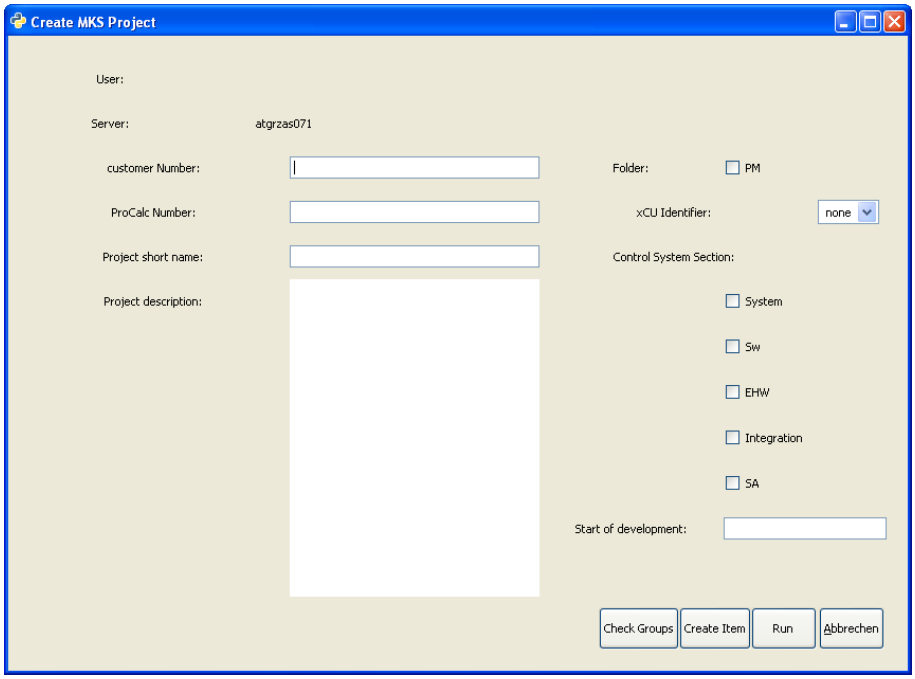

Figura 4.2: Finestra per la creazione di un nuovo progetto

cliente, e poi un codice alpha-numerico che indica il nome del progetto. Tutti questi codici e eventuali nomi vengono controllati da un apposito responsabile, che dovrebbe essere interpellato dagli utenti prima di compilare il modulo di richiesta, per avere, cosí, il codice ufficiale del progetto. Tutto questo, a volte, non accade, perció, in questi casi é necessario contattare l'utente per spiegare come ottentere un modulo corretto.

1.2 Cos'é il MAPSEC?

Prima di inviare la richiesta é necessario che il Team Leader crei gli appositi gruppi e aggiunga i componenti del suo team ad essi. Per fare ció alcuni utenti, i team leader, possono avere accesso ad una parte di MKS che permette questa operazione.

Per i vari team l'apertura di un nuovo progetto non é una cosa frequente, questo fa si che gli utenti si dimentichino come fare e cosa sia necessario. Questa parte crea dei problemi, infatti gli utenti, che usano poco frequentemente MKS o che non hanno fatto i training di introduzione all'uso di MKS, non hanno ben chiaro cosa siano i gruppi e i permessi. La conseguenza della mancata creazione dei gruppi genera l'impossibilitá di poter accedere al progetto creato, in quanto non si hanno i permessi per farlo.

Quando viene creato un progetto, infatti, vengono autorizzati solo i gruppi relativi a quel progetto, perció, se un utente non é inserito in uno di questi, non puó accedere o modificare i dati inseriti nel progetto. Affinché un utente possa avere la possibilitá di lavorare sul progetto é, quindi necessario che faccia parte di uno dei gruppi autorizzati e questo si puó fare solo grazie al Mapsec.

1.3 La non modificabilitá dell'Integrity Server

Come spiegato nel *Capitolo 1* il server Integrity contiente il vero e proprio progetto, quindi in esso sono salvati i dati piú delicati. Lo scorso anno é stato introdotto un altro server chiamato Staging Server. Questo nuovo server serve come ulteriore "protezione"all'Integrity Server, infatti per fare delle modifiche all'Integrity é necessario farlo prima nello Staging server e poi trasferire le modifiche nell'Integrity. Ovviamente questo non vale per le modifiche di routine, con questo intendo dire che per un utente normale le cose non sono cambiate molto, ma per un admin, che lavora nel sistema in maniera differente,le cose sono cambiate molto.

Non é piú possibile, quindi modificare direttamente i dati contenuti nell'Integrity server. Per quanto riguarda la creazione di nuovi progetti, questo si traduce nel fatto che per creare un nuovo progetto nell'Integrity server, é necessario farlo prima nello Staging Server e poi importarlo nell'Integrity Server.

Un problema che é sorto dopo l'introduzione dello Staging Server é stato che non é stata fornita la rispettiva riga di comando per fare l'operazione di import, quindi per rendere uguali i due server é necessario farlo solo dalla Admin GUI di MKS. Questa operazione sebben sia banale richiede molto tempo e svariati passaggi, ma questo non é risolvibile finché l'azienda che fornisce il software non fornisce la corrispondente comando.

1.4 Conferma creazione

Alla fine del processo di creazione ed dopo aver controllato che il progetto sia creato correttamente, cioé che esistano:

- progetto in MKS Source;
- progetto in MKS Integrity;
- i permessi sia correttamente settati per i vari gruppi;
- l'Item principale sia stato creato e i dati al suo interno siano corretti.

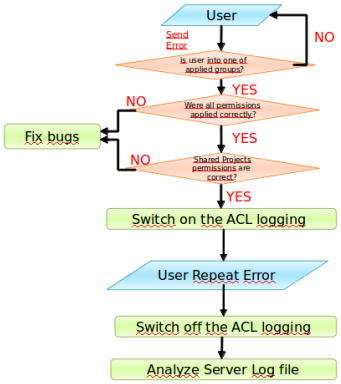

Figura 4.3: Schema per l'individuazione e soluzione di un problema dovuto ai permessi

L'ultimo passo é quello di informare l'utente che ha richiesto il progetto dell'avvenuta creazione e della posizione del suo progetto.

2 Errori dovuti ai Permessi

Prima dell'utilizzo di Soraja i permessi di accesso ad un progetto erano dati manualmente, essendo l'uomo molto meno preciso di una macchina, questo comportava molte disuguaglianze ed errori. Con l'avvio di Soraja tutti permessi si sono uniformati, quindi sono iniziati ad emergere tutta una serie di richieste di risoluzione di problemi da parte di molti utenti.

Inizilamente il problema principale é stato che molti utenti, nonostante fossero parte attiva dei team, non erano stati inseriti nei gruppi appropriati. Ció comportava l'impossibilitá per gli utenti di accedere al loro progetto. Arrivava, cosí, una richiesta del tipo "L'utente X non puó accedere al progetto". In questo caso la prima cosa da fare é controllare se l'utente X era parte dei gruppi consentiti per quel progetto. Se questa condizione non era soddisfatta bisognava contattare il Team Leader e chiedere che venga fatto correttamente il MAPSEC. Questo approccio ha risolto la maggior parte dei problemi, arrivando, cosí ad ottenere che tutti gli utenti fossero inseriti nei gruppi di appartennenza.

In un secondo momento ci siamo resi conto che i permessi standard che noi attivavamo per ogni progetto non erano sfruttati al meglio, perció é stata necessaria un'analisi dei vari possibili permessi attivabili, cercando di capire cosa realmente comportava l'attivazione o meno di questi. É stato trovato, cosí, un problema nella documentazione fornita da MKS, infatti mancava comple-

tamente la parte relativa ai permessi ed era, cosí, per noi impossibile capire le potenzialitá di questo strumento.

2.1 Controllo dei permessi

Come spiegato in precendenza la maggior parte delle richieste di aiuto veniva risolta scoprendo che l'utente richiedente non faceva parte dei gruppi autorizzati. Per rendere piú agevole questa ricerca, in Soraja, é stato creato un apposito strumento, il quale riportava per ogni utente un elenco di gruppi di appartenenza, quindi é facile individuare se il MAPSEC é stato compilato correttamente o meno. Ad esempio: se l'utente X lamenta il fatto di non poter accedere al progetto Y, attraverso questa finestra é immediato scoprire se X ha tra i suoi gruppi di appartenenza quello autorizzato nel progetto Y. Se il gruppo, relativo al progetto Y, non é presente nell'elenco, la semplice soluzione al problema é quella di dire all'utente di contattare il suo Team Leader e di invitarlo a fare correttamente il MAPSEC.

2.2 Uso del Log File

Se il problema, posto dall'utente, non é un problema di MAPSEC, cioé l'utente é correttamente inserito nei gruppi corretti, e non é un problema di permessi, cioé i permessi attivi sono tutti quelli sufficenti affinché l'utente possa lavorare, allora per individuare quale operazione crea il problema é necessario usare il Log File.

Il Log File, letteralmente *Diario di Bordo*, é un file in cui si registrano le attivitá compiute da un'applicazione, da un server o da un interprete, nel nostro caso da MKS. Normalmente il Log File di MKS non memorizza tutte le operazioni compiute, per fare in modo che ció accada, inviduando cosí il problema, é necessario attivare la registrazione. Attraverso l'interfaccia Admin di MKS é possibile attivare la registrazione solo per un determinato tipo di operazioni, in questo caso solo le operazioni che richiedono un permesso.

Dopo aver attivato la registrazione nel Log File, bisogna richiedere all'utente la ripetizione dell'operazione che generava l'errore. Accedendo poi al Log File é possibile individuare l'esatta operazione che ha generato l'errore, in quale file essa é incorsa, e ovviamente il nome dell'utente che ha richiesto quel tipo di operazione.
Dall'analisi del Log File si puó individuare piú precisamente il problema, e quindi capire chiaramente come risolverlo.

2.3 Leggere il problema

L'analisi degli errori attraverso il Log File, ci ha aiutato a capire il vero valore di ogni singolo permesso, ad esempio: il permesso chiamato *Accesso* indica l'accesso ad un progetto, cioé la visione del progetto, o accesso inteso come possibilitá anche di consultazione dei file contenuti nel progetto. Capendo, cosí, quali fossero i permessi piú opportuni per ogni tipo di utente.

Un altro problema che si é risolto, usando il Log File é stoto quello dei Progetti Condivisi. I Progetti Condivisi sono quei File che fanno parte di piú progetti, ad esempio: nello sviluppo di un nuovo software si puó usare una parte di codice giá scritta per un software precedente. Questo fatto implica che l'utente che lavora sul progetto principale deve avere i permessi anche per il progetto condiviso, altrimenti non potrá in alcun modo lavorare per il mioglioramento o per il semplice utilizzo di quest'ultimo.

Il Log File si é rivelato utile anche per alcune incompresioni riguardo ai Nickname. AVL fornisce ad ogni dipendente un nome utente formato dalle prime cinque lettere del cognome e la prima del nome. Sfortunatamente con l'ingrandirsi dell'azienda ci sono stata frenquenti casi di omonimi, in questi casi il secondo arrivato ha ottenuto un nickname scritto tutto in maiuscolo. Questa scelta, non tra le piú ottimali, ha creato non pochi disagi. Infatti essendo il nickname dell'azienda lo stesso usato in MKS, spesso i team leader inserivano nel MAPSEC l'utente sbagliato, confondendo il nickname scritto in maiuscolo con quello in minnuscolo.
Fortunatamente hanno deciso di cambiare metodo di assegnazione dei nickname, usando un codice alpha-numerico come Nickname, risolvendo cosí le incomprensioni.

Conclusioni

L'esperienza da me fatta é stata sicuramente positiva, l'autonomia nella gestione di alcuni tipi di richieste provenienti dagli utenti, mi ha dato modo di capire la difficoltá di creare un software il piú chiaro, allo stesso tempo efficente, possibile. L'approccio di molti utenti ad un nuovo software é molto difficile e mal vista, il capire come sfruttare al meglio uno strumento richiede tempo e spesso gli utenti, con cui ho avuto a che fare, non erano disposti a investirlo.
Molti utenti, costretti ad usare MKS, molte volte, invece di cercare di capire il problema da loro generato, mandavano direttamente una richiesta di aiuto a noi, quindi il lavoro era semplicemnte un spiegargli come usare il software, piú che risolvere veri e propri problemi.

Per quanto riguarda la parte di programmazione le GUI da me create sono ancora ampiamente in uso e in sviluppo. Il Python come linguaggio, a mio parere, risulta molto intuitivo e agile da scrivere e leggere. Ho capito come nella scrittura di un programma, spesso, la difficoltá sta nel commentare piú che nel programmare. Piú un codice é commentato, piú facile sará la sua comprensione da parte di altri programmatori, e quindi il suo riutilizzo, come la logica dei linguaggi di programmazione orientati ad oggetti vuole.

Bibliografia

[1] *Lutz M. e Ascher D.*, "Programmare in Python", Editore Ulrico Hoepli, Milano

[2] "What is GTK+, and how can I use it? ", `www.gtk.org`

[3] *John Finlay*, "Pygtk.2 Tutorial", `www.pygtk.org`

[4] `www.en.wikipedia.org/wiki/MKS_Integrity`

[5] `www.mks.com/platform/our-product`