Università degli studi di Padova

Facoltà di Scienze MM.FF.NN

Corso di Laurea triennale in Informatica

Tesi di laurea

Analisi, progettazione ed implementazione di un modulo software per la gestione di un piano principale di produzione

Relatore: Dr. Francesco Ranzato

Candidato: Massimiliano Rosso

ANNO ACCADEMICO 2007/2008

Indice generale

Indice figure

Sommario

Il presente documento ha, come obbiettivo, quello di illustrare nel dettaglio l'attività di stage svolta presso Madlab Srl di Peraga di Vigonza (PD).

L'azienda in questione, scelta tra quelle presenti nella vetrina dell'evento STAGE-IT 2008, proponeva l'implementazione di un nuovo modulo da collegare ad Automa (software gestionale prodotto, eventualmente personalizzato ed infine distribuito da Madlab) in seguito ad una specifica richiesta di uno dei propri clienti, Manitou Spa.

La funzionalità desiderata, era quella di permettere ad un operatore, responsabile della gestione del piano principale di produzione di Manitou, di effettuare degli spostamenti, su scala temporale, dei lotti di produzione distribuiti tra le diverse linee, senza dover aggiornare manualmente i campi riguardanti la data e l'ora di montaggio di ogni singolo articolo nella tabella dei dettagli relativi al lotto considerato.

Le prime settimane sono state dedicate all'apprendimento delle basi della logistica, delle diverse realtà in cui opera Automa ed allo studio di alcuni casi particolari di aziende che hanno scelto di adottare diverse personalizzazioni di questo software.

Da una visione generale del funzionamento del sistema, ci siamo focalizzati sul problema posto da Manitou, studiando il processo di produzione all'interno dello stabilimento talvolta anche dal vivo, per poter meglio tracciare una buona analisi dei requisiti.

Successivamente, l'attenzione si è spostata sui linguaggi con i quali è stato sviluppato Automa, ovvero Visual Basic 6.0 e T-SQL.

Una volta apprese tali basi, sono state analizzate le porzioni di codice strettamente collegate allo scopo finale dell'attività di stage.

Presa confidenza con l'ambiente di lavoro, è stato scelto lo strumento necessario alla creazione del nuovo modulo software ed il tempo per imparare ad utilizzarlo si è rivelato maggiore del previsto anche a causa della documentazione, piuttosto scarna, allegata allo stesso.

Le ultime quattro settimane di stage sono state spese per l'implementazione del modulo, al termine della quale si è passati alla gestione degli errori, ai perfezionamenti dell'interfaccia grafica, alla correzione e pulizia del codice ed infine ai test di verifica.

Capitolo 1

L'azienda e lo stage

1.1 L'azienda

Madlab Srl è una piccola azienda di Peraga di Vigonza (Padova) che, pur essendo nata da poco, vanta un personale ormai molto esperto in materia di logistica e sistemi per la gestione di magazzini e movimentazione interna. E' stata fondata, infatti, nel 2007 come conseguenza di un distaccamento di MPH Srl, azienda milanese che opera da molti anni nello stesso settore, fornendo ai propri clienti dispositivi elettronici quali stampanti a trasferimento termico per codici a barre ed identificatori automatici in radiofrequenza, necessari al tracciamento dei flussi di materiali.

Il rapporto di collaborazione tra le due aziende continua ad essere tale per cui gran parte dei clienti di una, coincidono con quelli dell'altra: Madlab personalizza per loro il software per gestire la realtà in cui operano, MPH fornisce agli stessi i dispositivi e l'assistenza necessari affinché il sistema funzioni nel migliore dei modi.

1.2 La logistica

Nonostante il concetto di logistica abbia origini antiche, la sua definizione, per come la conosciamo oggi, è cambiata solo nel dopoguerra, negli ultimi sessant'anni. Prima di allora, infatti, è sempre stata considerata una branca dell'arte militare che trattava tutte quelle attività volte ad assicurare agli eserciti quanto si rendesse necessario per vivere, muovere e combattere nelle migliori condizioni di efficienza.

Ampliata ed estesa in campo economico, la logistica oggi comprende l'insieme delle attività organizzative, gestionali e strategiche che

governano nell'azienda i flussi di materiali e delle relative informazioni dalle origini presso i fornitori fino alla consegna dei prodotti finiti ai clienti e al servizio post-vendita.

1.3 Il piano principale di produzione

Un piano principale di produzione (in inglese MPS - Master Production Schedule) è un programma di fabbricazione che quantifica processi significativi, compiti ed altre risorse allo scopo di ottimizzare la produzione, identificare i colli di bottiglia, ed anticipare bisogni e beni finiti.

Da quando un MPS guida molte attività industriali, la sua qualità influenza drammaticamente la redditività industriale.

Generalmente gli MPS sono creati da software con significativi input scelti dagli utenti. A causa delle limitazioni di tali software, ma specialmente dall'intenso lavoro richiesto dalla stesura degli MPS, questi non includono ogni possibile aspetto della produzione ma solo gli elementi chiave che hanno dimostrato di essere controllabili efficacemente, come per esempio le ore di lavoro, i macchinari, la disponibilità di magazzino e le componenti di scorta.

La scelta del modello da costruire varia tra le aziende e tra le fabbriche manifatturiere.

Un piano principale di produzione è una dichiarazione di cosa le società si aspettano di produrre e di acquistare espressi in articoli selezionati, quantità specifiche e dati.

L'MPS traduce il business plan, includendo la domanda prevista, in un piano di produzione usando ordini programmati in un vero e proprio componente opzionale multi-livello per ambiente di schedulazione.

L'utilizzo di un MPS aiuta ad evitare mancanze, costosi acceleramenti, imprevisti dell'ultimo minuto ed inefficienti impieghi delle risorse. Permette inoltre di consolidare le sezioni progettate, produrre programmi guida ed effettuare previsioni affidabili per ogni livello di distinta base per qualsiasi tipo di componente.

1.4 Il cliente

Manitou Spa (azienda francese la cui sede italiana è situata a Cavazzona di Castelfranco in provincia di Modena) è ormai un affermato leader mondiale nella produzione di macchine per la movimentazione fuoristrada di grandi quantità di materiali pesanti o voluminosi. Alcuni esempi di tali macchine possono essere muletti, gru, elevatori telescopici, trattori agricoli e molte altre.

Già da prima della nascita di Madlab, Manitou ha adottato Automa (quando i diritti di questo software erano detenuti da MPH) e, nonostante le opere di personalizzazione siano state davvero importanti, tutt'oggi Madlab spende molte delle proprie risorse per adempiere alle richieste del cliente in questione.

1.4.1 Struttura operativa di Manitou Spa

La merce, una volta ricevuta, viene scaricata in un'area temporanea, detta “acema”, nella quale viene successivamente smistata e trasportata direttamente in linea di produzione (nel caso in cui ve ne sia urgente bisogno) o in un'apposita cella del magazzino, tipicamente nei pressi di materiale dello stesso tipo.

L'area adibita alla produzione delle macchine è composta da diverse linee di produzione, a loro volta suddivise in varie postazioni.

Ad ogni linea non viene dato in input un comando di inizio produzione per ogni singola macchina (commessa) ordinata da un cliente della Manitou ma viene chiesto di produrre un insieme di *n* commesse (lotto di produzione).

Ogni postazione è specializzata in una particolare fase di montaggio che avvicina sempre più il semi-assemblato al prodotto finito. Esistono anche delle linee mono-postazione, dette “isole”, nelle quali vengono effettuati i cosiddetti “premontaggi”, ovvero parti complesse e tipicamente standardizzate per la maggior parte dei prodotti di Manitou, che necessitano di essere costruite in stock parallelamente al resto dei semi-

assemblati per poi essere unite a questi in una particolare fase del montaggio. Esempi di premontaggi possono essere il motore, la cabina di pilotaggio o un complesso braccio utensile comune a diversi modelli di macchinari.

Nonostante vi sia una ben determinata area adibita a magazzino, ogni settore dello stabilimento è considerato come una possibile zona di stoccaggio merci; questo per cercare di sfruttare al massimo lo spazio disponibile, ma soprattutto per poter monitorare ogni movimentazione di materiale in modo da sapere in ogni istante dove si trovi un determinato articolo.

Il magazzino vero e proprio è suddiviso in corsie, ognuna delle quali è composta a sua volta da diverse celle, identificate da codici a barre univoci, come qualsiasi articolo presente nello stabilimento. Esiste anche un magazzino automatico considerato come un'unica enorme cella, nel quale vengono stoccati alcuni tipi di materiali ingombranti.

Anche le singole linee di produzione hanno un'area riservata (opportunamente codificata), chiamata “bordo-linea” per le scorte di materiale utile alla produzione in corso: informaticamente parlando, ogni bordo-linea è una sorta di “buffer” che permette di evitare continui accessi al magazzino che comporterebbero un dispendio insostenibile di risorse fisiche, temporali ed umane.

Per inciso, qualsiasi movimento effettuato viene comunicato al sistema dandogli in input (tramite terminale ottico) il codice della cella o area di origine, il codice dell'articolo o del gruppo di articoli (pallet) da spostare ed, infine, la cella o area di destinazione della merce.

E' utile far notare che il funzionamento di tale sistema, è strettamente dipendente dalla corretta attività dei magazzinieri e degli operatori in genere. Se i primi non comunicassero ad Automa uno spostamento della merce (tramite i dispositivi di identificazione automatica) o se i secondi compissero un lavoro diverso da ciò che il sistema si aspetta (ad esempio il montaggio di un pezzo errato), si verificherebbero delle anomalie difficilmente gestibili in quanto la disponibilità e la locazione reale del materiale non corrisponderebbero più a quelle registrate nel database.

Dal momento che la linea di produzione avanza indipendentemente da

tutto il resto, è facile intuire che non sono tollerabili frequenti mancanze di materiale a bordo-linea in quanto, dalla linea di produzione a cui appartiene la postazione in deficit di materiale, uscirebbe un cosiddetto “mancante” ovvero un prodotto che dovrebbe essere finito ma che in realtà è incompleto. Sapere quale tipo di articolo abbia determinato tale anomalia e quale postazione non sia riuscita a portare a termine la propria fase lavorativa è indispensabile per poter andare, una volta recuperato il pezzo mancante, a rimediare a colpo sicuro l'incompletezza della macchina prodotta. E' proprio grazie ad Automa che situazioni di questo tipo, possono essere evitate o comunque risolte in breve tempo.

1.5 Automa

Il prodotto di Madlab Srl (precedentemente di proprietà di MPH Srl), battezzato Automa, nato come gestionale per lo stoccaggio di merci e preparazione di spedizioni, in seguito alle esigenze del mercato logistico, si è evoluto in un software per la gestione dei flussi di materiale all'interno di un magazzino o di una linea di produzione.

Madlab personalizza il proprio sorgente di volta in volta assecondando il più possibile le richieste del cliente, aiutando quest'ultimo ad ottimizzare la propria attività, tenendo traccia di qualunque cosa avvenga all'interno dell'area lavorativa presso la quale viene installato tale software.

Automa convive con il sistema informatico che il cliente già possiede e, a seconda della realtà in cui viene adattato, si trova a dover dialogare con altri programmi, per esempio con il database integrato in AS/400 (minicomputer sviluppato da IBM dal 1988 e adottato ancora oggi dalla maggior parte dei clienti di Madlab) oppure con software di controllo robotizzato di macchinari quali, ad esempio, linee di assemblaggio automatiche o trasloelevatori presenti in ogni magazzino verticale moderno.

In ogni caso, per rilevare e monitorare qualsiasi movimentazione di materiale effettuata da un operatore all'interno dell'area lavorativa, sono necessari dei dispositivi di identificazione automatica (ad esempio lettori

di codici a barre), ormai indispensabili all'interno di impianti che trattano grandi quantità di articoli di diverso tipo. Tali dispositivi sono tipicamente terminali portatili a radio frequenza e pilotati da un secondo software, anch'esso in simbiosi con Automa.

Da un punto di vista di più basso livello, Automa è un database (gestito, nella sua attuale versione, presso la maggior parte dei clienti di Madlab, Manitou compresa, dal DBMS Microsoft SQL server 2005) le cui tabelle riproducono fedelmente non solo la struttura fisica dello stabilimento nel quale viene installato ma anche la logica di gestione adottata dal cliente che lo acquista. Le interfacce presentate sulle varie workstation agli organizzatori dei lotti di produzione (o ai responsabili della suddivisione del lavoro in genere) ed ai magazzinieri sui diversi terminali wireless, sono implementate in Visual Basic 6.0. Tali interfacce pilotano, a seconda della programmazione dei lotti da mandare in produzione, le transazioni all'interno del database, richiamando opportune stored procedure e funzioni SQL che garantiscono l'integrità (quasi) totale dei dati.

Il fatto che Automa sia nato con uno scopo diverso da quello per cui viene utilizzato in Manitou, le molteplici e quotidiane modifiche al codice sorgente da parte di più persone per assecondare gli improvvisi cambiamenti nella politica di gestione del cliente, il riutilizzo di elaborati moduli creati per altri impianti, adattandoli alla realtà in questione, l'impossibilità di sconvolgere profondamente la struttura del database e la necessità di inserire una grande quantità di ridondanza di dati nelle tabelle per evitare di appesantire troppo il già immenso lavoro di lettura che il server SQL deve sopportare, sono le cause che hanno reso, nel corso degli anni, il codice di Automa prolisso e contorto.

1.6 Il problema

Descritta sommariamente e in modo molto semplificato la realtà in questione, è possibile parlare più chiaramente del mio ruolo in questo stage.

Negli ultimi tempi Manitou si è trovata ad avere la necessità di sconvolgere da un momento all'altro la pianificazione dei lotti accodati presso le diverse linee. Dal momento che Automa non prevede situazioni di questo tipo e non è in grado quindi di gestirle, attualmente è necessario sistemare i record del database "a mano".

Ovviamente, data l'enorme quantità di record trattati, è quasi impossibile in questo modo mantenere l'integrità dei dati e garantire la consistenza del database a fronte di un'operazione simile.

1.7 Lo stage

Madlab ha pensato di rimediare al problema descritto precedentemente tramite lo sviluppo di una nuova funzionalità di Automa che faciliti la pianificazione dell'ordinamento delle code di lotti di produzione e dello smistamento di questi tra le diverse linee di assemblaggio, ma soprattutto che permetta di effettuare uno "swap" tra lotti sulla linea temporale.

Si è pensato al risultato finale come ad uno strumento grafico molto simile ad un diagramma di Gantt il cui asse orizzontale rappresenti il tempo mentre quello verticale l'elenco delle linee di produzione. All'interno dello spazio limitato da tali assi, i lotti sono distribuiti in corrispondenza delle rispettive linee e visualizzati come oggetti rettangolari, la cui lunghezza ne rispecchia il tempo di occupazione delle risorse meccaniche ed umane.

Da tale interfaccia, si desidera, inoltre, poter effettuare le movimentazioni previste dall'aggiornamento del piano principale di produzione, semplicemente trascinando con il puntatore del mouse i vari lotti, posizionandoli nell'intervallo di tempo più consono alle attuali necessità dell'azienda.

A seguito di tale operazione, il database di Automa dev'essere aggiornato in automatico, garantendo la totale integrità dei dati e la coerenza degli stessi con i rispettivi articoli reali all'interno dello stabilimento, senza dover mettere mano alle tabelle tramite query di aggiornamento, inserimento o cancellazione.

Capitolo 2

Fase di apprendimento e formazione

Le prime tre settimane di stage sono state dedicate all'apprendimento delle conoscenze necessarie allo svolgimento dell'attività.

Il tutor aziendale ha cercato, prima di tutto, di farmi capire il concetto generale di logistica e di come Madlab ha scelto di affrontare il problema nelle diverse realtà in cui si presenta.

Ho avuto modo di visitare il nuovo stabilimento milanese di un importante cliente di Madlab, il quale aveva da poco installato nella propria sede il software Automa e, giusto in quei giorni, stava portando a termine il collaudo e la formazione del proprio personale all'utilizzo del nuovo sistema.

Data la semplicità della struttura in cui mi trovavo e grazie al fatto che il cliente in questione non aveva necessità di gestire linee di produzione ma piuttosto un magazzino automatico, è stata l'occasione giusta di approcciare con una personalizzazione di Automa meno complessa di quella che avrei dovuto affrontare a breve.

Nei giorni seguenti sono stato invitato a visitare entrambi i magazzini di Manitou. Dal momento che i meccanismi relativi allo stoccaggio merci mi erano già stati illustrati, l'attenzione è stata rivolta quasi esclusivamente alla catena di produzione. Grazie alle due visite presso Manitou mi è stato possibile capire, in poco tempo e senza alcuna ambiguità, la struttura operativa e la logica, precedentemente descritte, adottate dal cliente in questione.

A questo punto il tutor mi ha chiesto di provare a definire ed illustrare il modo in cui avrei gestito informaticamente una realtà simile a quella di Manitou (con ovvie semplificazioni). Il risultato è stato un diagramma UML minimale ma completo e formalmente corretto (non riportato nel presente documento di tesi in quanto utile solamente come esercizio), prova del fatto che avessi compreso le necessità ed i problemi del cliente.
Fatto ciò sono potuto passare allo studio degli strumenti di lavoro.

Inizialmente ho imparato a muovermi su SQL Server 2005, DBMS a me sconosciuto fino ad allora, interrogando i database di alcuni clienti di Madlab e, talvolta, utilizzando funzioni che esulano dal contesto relativo allo stage. In tal modo ho potuto dare un veloce ripasso ai principali costrutti di SQL che mi sarebbero serviti nella fase di sviluppo.

Per quanto riguarda lo studio di Visual Basic 6.0, ho fatto riferimento, oltre alla documentazione online, a varie guide trovate in Internet. Dopo essermi esercitato per prendere confidenza con tale linguaggio, con il quale non avevo mai avuto a che fare, ho analizzato il codice di alcune funzioni di Automa.

L'ultimo strumento, oggetto del mio studio, è stato il controllo OCX phGantTimePackage 3.0, descritto più avanti in questa stessa sezione. Questa parte della fase formativa ha occupato una notevole quantità del tempo a disposizione, sia per l'innumerevole numero di funzionalità integrate nell'OCX stesso, sia per il fatto che la documentazione ad esso allegata fosse, a mio avviso, relativamente scarsa e talvolta poco chiara.

2.1 Microsoft SQL Server

Microsoft SQL Server è un DBMS relazionale prodotto da Microsoft. Nelle prime versioni era utilizzato per basi dati medio-piccole, ma a partire dalla versione 2000 è stato utilizzato anche per la gestione di basi dati di grandi dimensioni. Usa una variante del linguaggio SQL standard (lo standard ISO certificato nel 1992) chiamata T-SQL Transact-SQL.

La versione di Microsoft SQL Server attualmente adottata da Madlab è la 2005.

2.2 SQL

SQL (Structured Query Language) è un linguaggio creato per l'accesso a informazioni memorizzate nei database. Nasce nel 1974 ad opera di

Donald Chamberlin, nei laboratori dell'IBM, come strumento per lavorare con database che seguano il modello entità-relazioni (E/R).

Essendo un linguaggio dichiarativo, SQL non richiede la stesura di sequenze di operazioni (come ad es. i Linguaggi imperativi), piuttosto di specificare le proprietà logiche delle informazioni ricercate. Esso si divide in tre sottoinsiemi:

DDL (data definition language - linguaggio di definizione dei dati) serve a creare, modificare o eliminare gli oggetti in un database. Sono i comandi DDL a definire la struttura del database e quindi dei dati ivi contenuti. Ma non fornisce gli strumenti per modificare i dati stessi: per tale scopo si usa il DML. L'utente deve avere i permessi necessari per agire sulla struttura del database e questi permessi vengono assegnati tramite il DCL.

DML (data manipulation language - linguaggio di manipolazione dei dati) fornisce i comandi per inserire, modificare, eliminare o leggere i dati all'interno delle tabelle di un database. La struttura di questi dati deve già essere stata definita tramite il DDL. Inoltre, il permesso di accedere a tali dati deve essere assegnato all'utente tramite il DCL.

DCL (data control language – linguaggio di controllo sui dati) serve a fornire o revocare agli utenti i permessi necessari per poter utilizzare i comandi DML e DDL, oltre agli stessi comandi DCL (che gli servono per poter a sua volta modificare i permessi su alcuni oggetti).

2.3 Transact-SQL

Transact-SQL (T-SQL) è l'estensione proprietaria del linguaggio SQL sviluppata da Microsoft e Sybase. La versione di Microsoft viene fornita insieme a Microsoft SQL Server.

Transact-SQL espande le prestazioni di SQL aggiungendo:

- Funzioni per controllo di flusso
- Possibilità di definire variabili locali.
- Varie funzioni per la manipolazione di stringhe, date, espressioni matematiche.

- Ampliamento delle istruzioni DELETE e UPDATE.

T-SQL è il linguaggio adottato da Madlab per creare e gestire il database di Automa.

2.4 Visual Basic 6.0

Il Visual Basic (formalmente abbreviato VB) è un linguaggio di programmazione *event driven*, la cui sintassi deriva dal BASIC.
Sviluppato dalla Microsoft, il Visual Basic è particolarmente noto per:

- la sua semplicità d'uso (non utilizza formalità di punteggiatura tipica di quasi tutti gli altri linguaggi);
- il suo ambiente di lavoro che permette di realizzare in breve tempo interfacce GUI anche complesse;
- il pratico accesso alle basi dati;
- la creazione di controlli ActiveX con il linguaggio stesso (nelle precedenti versioni si doveva usare il linguaggio C).

Tramite l'integrazione di alcuni tipi di controlli, ad esempio gli OCX, presenti nelle versioni a 32 bit e talvolta realizzati da altri programmatori, è possibile ampliare le potenzialità del linguaggio, aggiungendo nei propri progetti nuove funzioni o funzioni più complete.
Il Visual Basic è stato (e lo è tuttora, anche se sta venendo lentamente soppiantato da Visual Basic .NET) uno dei linguaggi più utilizzati al mondo. Il Visual Basic ha creato il primo mercato commerciale di componenti riutilizzabili. esistono migliaia di componenti di terze parti disponibili per gli sviluppatori. Un esempio di tali componenti è proprio il controllo OCX da me utilizzato durante lo svolgimento dell'attività di stage. Il Visual Basic rende semplice sviluppare e riutilizzare componenti.

Visual Basic viene utilizzato da Madlab per la creazione delle interfacce grafiche e dello strato software posizionato tra queste ed il database di Automa. La versione attualmente adottata dall'azienda è la 6.0. Nonostante sia una versione ormai vecchia e non più supportata da Microsoft dal 1998, riscrivere per intero Automa utilizzando un linguaggio più recente sarebbe un'operazione impensabile con le attuali risorse aziendali.

2.5 Il controllo phGantTimePackage

Per lo sviluppo del modulo, mi sono servito di un componente OCX (OLE Control eXtension), ovvero un controllo che estende le funzionalità di base di Visual Basic. Questo controllo, chiamato phGantTimePackage, prodotto e distribuito dalla software house svedese PlexityHide, è una libreria di funzioni utili ad integrare nel progetto in sviluppo la possibilità di visualizzare e manipolare diagrammi grafici temporali, quali, ad esempio, i diagrammi di Gantt.

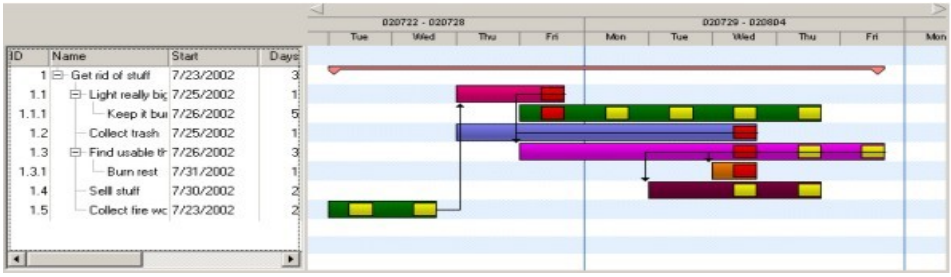

Figura 2.1 - Esempio di utilizzo di phGantTimePackage

Una volta importato l'OCX nell'ambiente di sviluppo e creata un'istanza di questo nella finestra in cui si sta lavorando, ciò che si ha a disposizione è una struttura in grado di visualizzare, all'interno di un riquadro grafico, infiniti oggetti, la cui caratteristica principale è la durata definita e limitata nel tempo. Aggiungendo il codice opportuno agli eventi “lanciabili” da questo componente, è possibile fare in modo di modificare le caratteristiche di tali oggetti, ad esempio l'occupazione di risorse, le relazioni tra due o più elementi, la durata degli stessi, l'istante di inizio, quello di fine e molte altre.

Capitolo 3

Analisi dei requisiti

3.1 Tracciamento

I requisiti del cliente sono stati analizzati e definiti tramite ripetitivi colloqui con il tutor aziendale e, talvolta con il cliente stesso. Alcuni di questi si sono resi evidenti solo durante lo sviluppo dell'applicazione.
In questa sezione verranno elencati i requisiti, adeguatamente suddivisi per categoria. Successivamente verranno illustrati e descritti in dettaglio gli use-case a dimostrazione del modo in cui l'utente e il sistema interagiscono tra loro. Infine verrà descritto un diagramma delle attività per rendere più chiaro il flusso standard degli eventi durante una normale esecuzione dell'applicativo.

3.2 Elenco dei requisiti

I requisiti sono stati catalogati tenendo conto del fatto che si distinguono tra funzionali o qualitativi, ovvero tra i requisiti il cui soddisfacimento è essenziale affinché il modulo sia utilizzabile e tra quelli che offrono semplicemente un valore aggiunto al prodotto finale. Sono stati inoltre suddivisi tra requisiti che fanno capo al sistema e quelli che riguardano invece l'utente.

3.2.1 Notazioni

La notazione x._ indica il requisito numero x.
La notazione _.Q indica un requisito di qualità.
La notazione _.F indica un requisito funzionale.

La notazione _._.s indica un requisito riguardante il sistema.
La notazione _._.s indica un requisito riguardante l'utente.

3.2.2 Requisiti

- **0.F.s** I dati relativi ai lotti accodati per la produzione devono essere modificabili direttamente tramite la nuova interfaccia.

- **1.F.s** Le modifiche effettuate tramite il modulo devono riguardare esclusivamente le date di produzione dei lotti considerati e di tutte le commesse che ne fanno parte.

- **2.F.s** Nel caso in cui un lotto sia distribuito su più linee di produzione, il trascinamento di questo sull'asse temporale, relativamente ad una linea, non deve influenzare le porzioni dello stesso, sulle altre linee.

- **3.F.s** L'utente deve poter riprendere il lavoro svolto fino a quel momento nonostante non abbia ancora deciso di rendere effettive le modifiche a livello di database.

- **4.F.s** Gli errori causati da malfunzionamenti della rete o da crash improvvisi del sistema non devono compromettere l'integrità dei dati.

- **5.F.s** Dev'essere tenuto conto del fatto che nei giorni lavorativi possono esserci degli intervalli di tempo in cui una o più linee vengono disattivate per manutenzione

- **6.Q.s** Il modulo deve tenere traccia, tramite un'opportuna tabella di log, di ogni dettaglio relativo a qualsiasi operazione che modifichi lo stato del database, dando la possibilità all'amministratore di sistema di risalire all'utente responsabile ed avere modo di ripristinare

facilmente la coerenza ed integrità dei dati.

- **7.Q.s** Gli errori causati da malfunzionamenti della rete o da crash improvvisi del sistema devono essere registrati in un apposito file di log.

- **8.Q.s** Inaspettate interruzioni della connessione di rete o crash improvvisi del sistema non devono far perdere all'utente il lavoro effettuato tramite il nuovo modulo.

- **9.Q.s** Dev'essere tenuto conto del fatto che nei giorni festivi e durante il fine settimana, tutte le linee di produzione dello stabilimento non sono operative.

- **10.Q.u** L'utente deve poter decidere, al caricamento dei dati, se ripristinare la situazione su cui aveva lavorato precedentemente, o quella relativa allo stato attuale delle linee di produzione.

- **11.Q.u** L'utente che stia effettuando operazioni delicate, che modifichino lo stato del database, dev'essere opportunamente avvisato, dandogli modo di annullare l'operazione in corso.

- **12.Q.u** L'utente deve poter effettuare la connessione e la disconnessione del modulo dal database in qualsiasi momento.

- **13.Q.u** L'utente deve poter visualizzare l'interfaccia a schermo intero o personalizzarne le dimensioni.

- **14.Q.u** Durante il funzionamento del modulo, nei momenti in cui vengono effettuate operazioni che impieghino un considerevole lasso di tempo, l'utente deve potersi rendere conto di ciò, in modo da capire che il sistema non è bloccato o per intercettare facilmente eventuali errori di qualsiasi natura.

- **15.Q.u** Le informazioni essenziali relative ai lotti visualizzati devono essere facilmente reperibili attraverso l'interfaccia per rendere più chiara la situazione attuale e ciò che si sta facendo.

- **16.Q.u** L'interfaccia deve rispondere ai comandi dell'utente nel più breve tempo possibile.

- **17.Q.u** I lotti devono essere facilmente distinguibili tra loro tramite diverse colorazioni.

- **18.Q.u** La visualizzazione dell'intervallo temporale deve poter essere resettata per dare all'utente un punto di riferimento nel caso in cui si sia allontanato troppo dalla data attuale.

- **19.Q.u** La visualizzazione della struttura dello stabilimento dev'essere personalizzabile dall'utente per agevolarlo nel proprio lavoro.

- **20.Q.u** L'intervallo di tempo visualizzato nell'interfaccia del modulo dev'essere personalizzabile dall'utente per agevolarlo nel proprio lavoro.

- **21.Q.u** Le funzionalità messe a disposizione dal modulo devono essere facilmente intuibili ed utilizzabili.

3.3 Descrizione degli use-case

3.3.1 Use-case A: accesso alle funzionalità del modulo

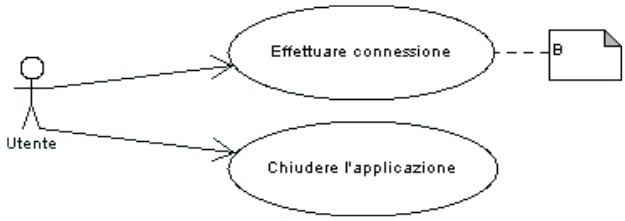

Figura 3.1 - Use-case A

Attori: utente

Scopo: stabilire una connessione tra il modulo sviluppato ed il database di Automa

Precondizioni: la rete è disponibile e l'utente ha avviato l'applicazione

Flusso base di eventi:

- ➔ l'utente clicca il pulsante di connessione

Flussi alternativi:

- ➔ l'utente chiude l'applicazione senza aver effettuato alcuna operazione

Post-condizioni: l'utente deve aver premuto il pulsante di connessione, la quale dev'essere stata stabilita senza alcun errore

3.3.2 Use-case B: caricamento e visualizzazione dei lotti

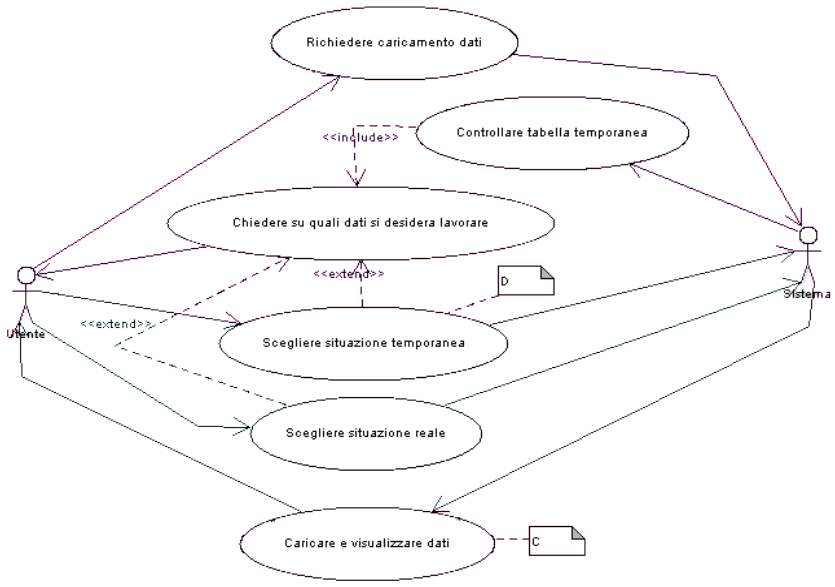

Figura 3.2 - Use-case B

Attori: utente, sistema

Scopo: caricamento dei dati relativi ai lotti di produzione e visualizzazione di questi tramite l'interfaccia

Precondizioni: dev'essere stata stabilita la connessione con il database di Automa

Flusso base di eventi:

- ➔ L'utente preme il pulsante relativo al caricamento dei dati presenti nel database
- ➔ Il modulo di gestione dei lotti verifica la presenza di record nella tabella temporanea che facciano riferimento all'utente connesso
- ➔ Nel caso in cui vengano trovati record che soddisfino i criteri di ricerca, viene chiesto all'utente se desidera caricare i dati presenti

nella tabella od ignorarli e caricare quindi i dati che rispecchino la reale situazione delle linee produttive

- ➔ L'utente sceglie la seconda tra le due possibilità
- ➔ Il sistema visualizza la struttura dello stabilimento sotto forma d'albero e, in corrispondenza di ogni linea, visualizza tutti i lotti accodati, considerando i dati "reali"

Flussi alternativi:

- ➔ L'utente decide di caricare i dati presenti nella tabella temporanea
- ➔ Il sistema visualizza la struttura dello stabilimento sotto forma d'albero e, in corrispondenza di ogni linea, visualizza tutti i lotti accodati, ripristinando l'ultima situazione elaborata dallo stesso utente e non ancora salvata
- ➔ L'utente può decidere di effettuare altri spostamenti o di salvare la situazione che aveva precedentemente impostato

Post-condizioni: indipendentemente dalla scelta effettuata dall'utente, l'interfaccia deve visualizzare correttamente i lotti suddividendoli tra le diverse linee alle quali sono assegnati

3.3.3 Use-case C: elaborazione del piano principale di produzione

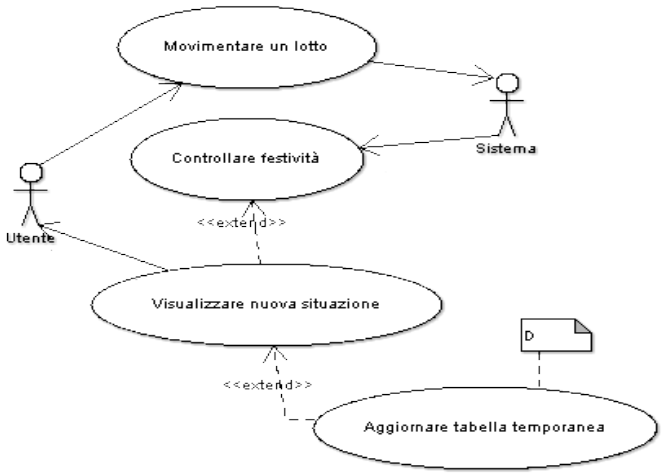

Figura 3.3 - Use-case C

Attori: utente, sistema

Scopo: elaborare il piano principale di produzione fino ad ottenere la situazione desiderata

Precondizioni: devono essere stati caricati i dati relativi alla situazione attuale o temporanea

Flusso base di eventi:

- Dopo aver scorso il grafico lungo l'asse temporale (orizzontalmente) ed avendo scelto la linea sulla quale operare cambiamenti (scorrendo verticalmente l'albero raffigurante la struttura dello stabilimento), l'utente trascina il lotto selezionato fino a posizionarne l'inizio nella data desiderata
- Il sistema verifica che la data scelta come inizio di produzione del lotto considerato, sia un giorno valido
- Nel caso in cui il giorno di inizio sia valido, il lotto viene posizionato nelle coordinate scelte ed il sistema controlla, iterativamente, fino alla data di uscita di produzione del lotto, la presenza di eventuali giorni non lavorativi per la linea in considerazione
- Per ogni data non valida incontrata durante tale controllo, la fine del lotto viene posticipata di un giorno, dando modo all'utente di rendersi conto della reale situazione a fronte di tale cambiamento
- Il record nella tabella temporanea, relativo al lotto movimentato, in corrispondenza della linea di produzione scelta dall'utente, viene aggiornato salvando le nuove date di inizio/fine produzione

Flussi alternativi:

- Una volta trascinato il lotto desiderato, nel caso in cui il giorno scelto come inizio sia festivo o di manutenzione per la linea considerata, il sistema sceglie la prima data valida successiva a tale giorno

Post-condizioni: L'utente deve aver effettuato almeno uno spostamento con successo

3.3.4 Use-case D: salvataggio della situazione rappresentata

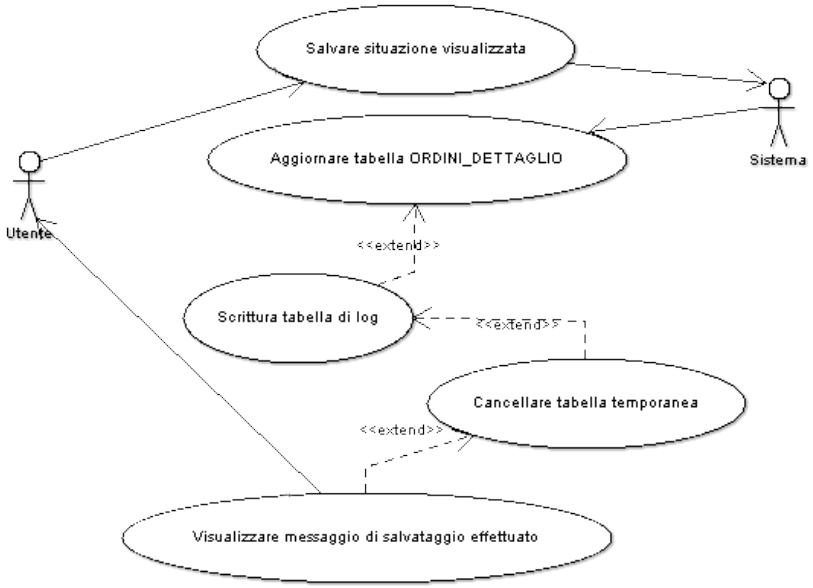

Figura 3.4 - Use-case D

Attori: utente, sistema

Scopo: rendere effettiva la situazione impostata dall'utente tramite l'interfaccia di gestione del piano principale di produzione

Precondizioni: l'utente deve aver effettuato lo spostamento di almeno un lotto o deve aver caricato la situazione presente nella tabella temporanea

Flusso base di eventi:

- ➔ L'utente preme il pulsante di salvataggio

- ➔ Il sistema, dopo aver chiesto conferma, effettua un aggiornamento massivo reimpostando tutte le date di produzione delle componenti associate ai lotti modificati, in base alle decisioni dell'utente
- ➔ Per ogni lotto spostato dall'utente prima del salvataggio, viene creato un record nella tabella di log, in modo da rintracciare ogni operazione delicata ed il rispettivo responsabile
- ➔ Ogni riga della tabella temporanea, relativa all'utente connesso, viene eliminata

Flussi alternativi: Nessuno

Post-condizioni: Le tabelle del database devono essere state correttamente aggiornate

3.4 Diagramma delle attività

3.4.1 Esecuzione standard del modulo

Il diagramma delle attività visualizzato qui di seguito, rappresenta i principali stati in cui il sistema si può trovare, a seguito di un utilizzo standard dell'applicazione sviluppata in sede di stage.
Sono state date per scontate le attività accessibili ad ogni istante (o quasi) dell'esecuzione, ad esempio, le funzioni di personalizzazione della visualizzazione, la connessione/disconnessione dal database o il caricamento dei dati (che sovrascrive la visualizzazione attuale senza renderla effettiva pur mantenendola salvata nella tabella temporanea).

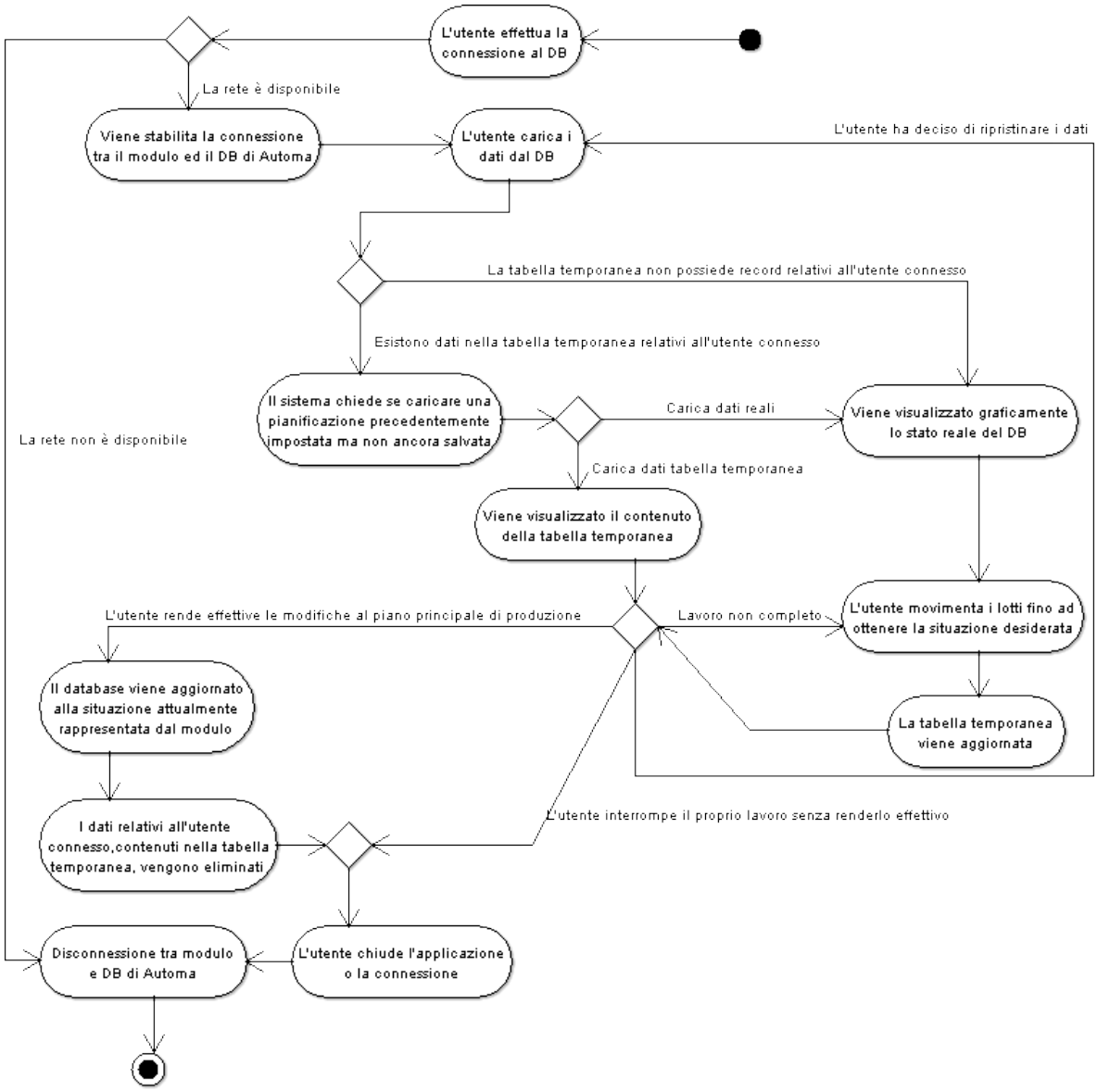

Figura 3.5 - Diagramma delle attività

Capitolo 4

Progettazione

In questa sezione saranno descritte le scelte progettuali rese poi concrete nella fase di sviluppo. Non è stato necessario apportare alcuna modifica alle tabelle esistenti nel database. Le uniche alterazioni di questo riguardano l'aggiunta di due tabelle e due funzioni, tutte descritte più avanti in questo capitolo.

4.1 I dati importanti

Nonostante l'imponenza del codice di Automa e l'elevato numero di tabelle del suo database e dei campi in esse contenuti, una volta capita la logica del problema ed appresi gli strumenti di lavoro, i dati che il modulo aggiuntivo deve acquisire, necessari al suo funzionamento, non sono molti.

Dal momento che, ad oggi, non esiste documentazione relativa ad Automa, e data l'impossibilità di schematizzare, nel tempo a disposizione, le relazioni presenti nel database su cui poggia l'applicazione, mi limiterò ad elencare le tabelle ed i campi strettamente correlati con il lavoro da me svolto, descrivendo le principali caratteristiche di ognuno di questi elementi.

4.1.1 La tabella Ordini_Testata

Come spiegato nella parte introduttiva di questo documento di tesi, Manitou manda in produzione dei lotti, i quali possono essere costituiti da una o più commesse. Ogni riga della tabella Ordini_Testata, fa riferimento ad una commessa di un cliente.

Figura 4.1 - Struttura tabella Ordini_Testata

I campi elencati sono solo una piccolissima parte di quelli realmente appartenenti alla tabella. Per motivi di spazio e, dato che tutti gli elementi omessi esulano dal contesto trattato nel presente documento, ho fatto riferimento ai soli attributi con cui ho lavorato. Tra questi, i primi sei costituiscono la chiave primaria di Ordini_Testata.

Per un'analisi più completa e dettagliata della tabella si faccia riferimento all'appendice in fondo al presente documento.

4.1.2 La tabella Ordini_Dettaglio

Ogni commessa mandata in produzione necessita di migliaia di componenti che, montati tra loro, danno vita al prodotto finito. Ogni record della tabella Ordini_Dettaglio rappresenta uno di tali componenti descrivendone lo stato, la posizione all'interno dello stabilimento, la data di montaggio e molte altre caratteristiche.

Figura 4.2 - Struttura tabella Ordini_Dettaglio

Come per la tabella precedente, ho dovuto limitare la descrizione (in appendice) dei campi ai soli utilizzati durante lo stage. I primi sei, oltre a costituire la chiave primaria della tabella assieme al campo rigaOrdi, fungono anche da chiave esterna, stabilendo una relazione uno a molti rispettivamente tra Ordini_Testata ed Ordini_Dettaglio. RigaOrd serve in pratica a distinguere tra loro le componenti appartenenti ad una stessa commessa.

4.1.3 La tabella Viaggi

La produzione, all'interno di Manitou, viene intesa solamente come insieme di commesse ordinate da uno o più clienti. Tale insieme prende il nome di lotto di produzione. Se invece di ordini di produzione si parlasse di ordini di trasporto, le commesse potrebbero essere intese come gli articoli presenti in magazzino ordinati dai clienti che ne necessitano una fornitura. L'insieme di commesse, a questo punto, potrebbe contenere tutti gli ordini di un gruppo di clienti ad esempio con sedi vicine tra loro, il trasporto delle quali potrebbe quindi essere effettuato con un unico

viaggio. Questa tabella è un chiaro esempio del fatto che Automa non fosse nato per gestire strutture complesse come quella di Manitou. Il significato della tabella Viaggi è stato riadattato, di volta in volta, al contesto in cui ci si trovava. In questo caso un viaggio corrisponde ad un lotto di produzione.

Figura 4.3 - Struttura tabella Viaggi

Anche questa volta mi sono limitato ai campi essenziali. Tutti quelli elencati e descritti (in appendice), costituiscono la chiave primaria della tabella.

La tabella in esame è collegata a molte altre all'interno del database ma l'unica relazione utile al mio scopo è quella stabilita con Ordini_Testata. Essendo quest'ultima legata direttamente ad Ordini_Dettaglio, è facile, tramite opportune query, trovare le associazioni tra le componenti di una determinata commessa ed il viaggio a cui questa appartiene.

4.1.4 La tabella Giorni_Non_Lavorativi

Tutti i giorni, al di fuori del fine settimana, per cui è prevista la sospensione dell'attività produttiva di una o più linee, sono inseriti in un'apposita tabella. Una linea può rimanere non attiva in una particolare data o a causa di una festività oppure per interventi di manutenzione o riparazione sulle postazioni di tale linea.

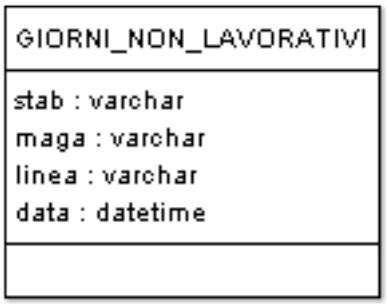

Figura 4.4 - Struttura tabella Giorni_Non_Lavorativi

Tutti i campi appartenenti alla tabella sono stati elencati (e descritti in appendice) e l'insieme di questi costituisce la chiave primaria di Giorni_Non_Lavorativi. Nel caso in cui vi siano periodi più lunghi di un giorno, durante i quali avvengono interventi sulle linee o semplicemente lo stabilimento rimane chiuso, nella tabella verrà inserito un record per ognuna delle date appartenenti agli intervalli di tempo considerati.

Un discorso simile dev'essere fatto per quanto riguarda il caso in cui due o più linee rimangano inattive nello stesso giorno. Una situazione di questo tipo prevede l'inserimento di un record per ogni linea che dovrà essere disattivata in quel particolare giorno.

4.2 Modifiche al database

Fin dall'inizio dell'attività di stage non era prevista alcuna alterazione alla struttura del database e così è stato se non per l'aggiunta di due piccole tabelle di appoggio.

Una delle due serve come supporto per i dati caricati una volta lanciata l'applicazione. Fino alla conferma, da parte dell'utente, che le modifiche al piano principale di produzione sono quelle definitive e devono quindi sovrascrivere i dati presenti nel database, questa tabella mantiene salvata la situazione su cui l'utente sta lavorando. Ad ogni modifica del grafico

visualizzato tramite l'interfaccia, la tabella viene opportunamente aggiornata. Progettando in questo modo l'applicazione, si garantisce il ripristino completo di tutto il lavoro svolto dall'utente fino al momento di un eventuale crash di sistema.

La seconda semplicemente registra i dettagli relativi ad ogni operazione delicata effettuata da un utente tramite il modulo aggiuntivo, in modo da rintracciare e risolvere eventuali problemi causati da un cattivo utilizzo del modulo, rendendo possibile risalire al responsabile del danno.

4.2.1 La tabella TMP_Gestione_Lotti

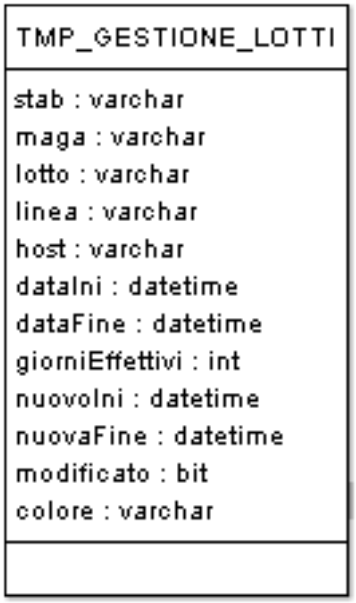

Figura 4.5 - Struttura tabella TMP_Gestione_Lotti

La prima delle due tabelle che ho aggiunto al database di Automa viene popolata direttamente dall'applicazione una volta premuto il pulsante di caricamento dei dati. Tutti i record salvati vengono associati al nome della workstation dalla quale l'utente ha avviato l'applicazione. In questo modo

la tabella funge da supporto per più connessioni del modulo da diverse postazioni, mantenendo i dati separati. Ogni record rappresenta un lotto produttivo.

Tutti gli altri attributi, ad eccezione di *host*, rappresentano le caratteristiche principali dei lotti. Parte dei campi della tabella non saranno mai a *null*, in particolare quelli che vengono inizializzati al caricamento dei dati. I campi che, invece, registrano i cambiamenti effettuati dall'utente, rimarranno a *null* fino a quando non saranno oggetto di operazioni di spostamento. L'insieme dei primi cinque attributi costituisce la chiave primaria di TMP_Gestione_Lotti.

Nell'appendice del presente documento è disponibile la descrizione in dettaglio di tutti gli attributi della tabella.

4.2.2 La tabella Log_Gestione_Lotti

Nonostante il log relativo ad un'applicazione possa essere salvato in un semplice file di testo (come spesso accade), registrare le operazioni delicate in una tabella contenuta nel database sotto controllo è una soluzione che rende più facile il filtraggio e la lettura delle informazioni in essa contenute. Solitamente un approccio del genere viene scartato principalmente per il fatto che, a lungo termine, genera grossi sprechi di spazio su disco. Il motivo per cui ho scelto di seguire questa strada, sta nel fatto che il modulo da me progettato ed implementato, è stato pensato per facilitare operazioni che vengono effettuate molto raramente, motivo per cui, la tabella Log_Gestione_Lotti difficilmente assumerà dimensioni eccessive, anche a distanza di anni. Se comunque ciò dovesse accadere, un'operazione di pulizia della tabella, eventualmente accompagnata dall'esportazione del contenuto della stessa su file di testo, sarebbe eseguibile in qualsiasi momento con uno sforzo irrisorio.

Figura 4.6 - Struttura tabella Log_Gestione_Lotti

Molti dei campi presenti in questa tabella sono già stati elencati precedentemente, d'altra parte, tramite il registro log non vengono introdotte informazioni funzionali al programma. All'occorrenza di ogni operazione che modifichi il database, eseguita tramite il nuovo modulo progettato, vengono salvate le informazioni più significative.

Per una descrizione più approfondita dei campi di tale tabella, si consulti l'appendice del presente documento.

4.3 I controlli principali

Per quanto riguarda la parte relativa al Visual Basic, i principali controlli, necessari allo sviluppo dell'interfaccia, sono cinque:

- **btnConn** è il pulsante che gestisce la connessione/disconnessione del modulo dal database

- **btnLoad** è il pulsante che permette di effettuare il caricamento dei dati dal database, al termine del quale vengono visualizzati la struttura dello stabilimento ed i lotti accodati presso le diverse linee

- **btnSave** è il pulsante che da il via alla sovrascrittura dei dati presenti nel database dopo che l'utente ha movimentato i lotti sullo schema grafico fino ad arrivare alla situazione scelta come definitiva

- **Form** è la finestra nella quale risiedono tutti gli altri controlli

- **Gantt** è il controllo di tipo phGantX, aggiunto ai controlli di base di Visual Basic 6.0 tramite l'OCX phGantTimePackage.

Una descrizione più dettagliata di tali componenti è riportata nell'appendice in fondo al presente documento.

Capitolo 5

Sviluppo

5.1 Le fasi di programmazione

Una volta terminata la fase di progettazione e scelte quindi la struttura delle varie componenti e la logica che le avrebbero legate, ho potuto dedicarmi all'implementazione vera e propria del modulo. La creazione di questo può essere descritta suddividendola in parti:

- aggiunta delle tabelle TMP_Gestione_Lotti e Log_Gestione_Lotti al database;
- realizzazione delle funzioni per il conteggio dei giorni effettivi di ogni lotto e per il calcolo della nuova dataProd da assegnare ad ogni record di Ordini_Dettaglio coinvolto nello spostamento di un lotto di produzione;
- realizzazione del form in Visual Basic e di tutte le componenti al suo interno;
- definizione delle funzioni ausiliarie, scritte in Visual Basic, invocate durante l'esecuzione del corpo di un determinato evento;
- implementazione degli eventi ridefiniti.

Dal momento che la struttura delle tabelle aggiunte al sistema esistente è già stata descritta nel precedente capitolo, la prima di queste fasi è di interesse relativo. Verranno pertanto trattate solamente le altre quattro parti riguardanti lo sviluppo.

5.2 Le funzioni in T-SQL

5.2.1 La funzione fnContGG

La prima delle due funzioni che ho dovuto aggiungere al database di Automa è *contGG*. Ha come parametri le date di inizio e fine del lotto considerato e, per aumentare l'efficienza di esecuzione, le coordinate *stab*, *maga* e *linea* a cui questo è stato assegnato. Viene richiamata durante il caricamento dei dati in TMP_Gestione_Lotti per aggiornare il campo *giorniEffettivi* in modo che contenga il numero di giorni lavorativi impegnati dal lotto descritto nel record corrente. Tra la data di inizio di un certo lotto e la data di fine dello stesso, infatti, vengono presi in considerazione tutti i giorni, compresi quelli festivi, i week-end ed eventualmente quelli in cui la linea di produzione assegnata al lotto è inattiva per lavori di manutenzione. Avere sempre a disposizione il numero dei giorni effettivi di un lotto è indispensabile in quanto, dopo un'ipotetica movimentazione dello stesso tramite il modulo da me creato, la visualizzazione di questo potrebbe cambiare a seconda dell'intervallo di tempo su cui verrebbe posizionato. Se, per esempio, prima di essere spostato, il lotto fosse a cavallo di un certo numero di giorni non validi e, dopo l'intervento dell'utente, il periodo compreso tra la nuova data di inizio e la nuova data di fine coinvolgesse un numero di giorni non validi diverso dal precedente, la dimensione dell'oggetto rappresentante il lotto aumenterebbe o diminuirebbe, in modo da dare informazioni immediate in merito ai tempi di occupazione delle linee, a fronte della scelta effettuata.

Perché tale operazione risulti corretta, una volta scelta la nuova data di inizio per un certo lotto, provvisoriamente viene scelta una nuova data di fine corrispondente alla somma tra la nuova data di inizio e il numero di giorni effettivi. A questo punto la funzione compie un numero di iterazioni pari al numero di giorni effettivi, controllando la validità delle date coinvolte giorno per giorno, partendo dalla nuova data di inizio fino alla nuova data di fine provvisoria. Ogni volta che si incontri un sabato, una domenica o un qualsiasi giorno non lavorativo, per la linea di montaggio

assegnata al lotto considerato, la nuova data di fine provvisoria viene incrementata di 24 ore. Il valore assunto da quest'ultima, una volta completato il ciclo di iterazioni, corrisponde all'effettiva nuova data di fine del lotto movimentato.

5.2.2 La funzione fnCalcolaNuovaDataProd

La seconda funzione scritta interamente in T-SQL si occupa di calcolare la nuova data di immissione in linea di un particolare componente di montaggio. Il valore ritornato da questa viene utilizzato per aggiornare il campo *dataProd* di tutti i record della tabella Ordini_Dettaglio che facciano riferimento alle commesse contenute nei lotti movimentati dall'utente. Tale funzione, infatti, viene invocata solamente al salvataggio del nuovo piano principale di produzione impostato dall'utente ed ha come parametri la data di inizio del lotto di cui il componente fa parte, il valore di *dataProd* attuale, la nuova data di inizio e, per aumentare l'efficienza di esecuzione, le coordinate *stab*, *maga* e *linea* in cui il componente dev'essere montato.

La funzione fnCalcolaNuovaDataProd prende in considerazione il numero di giorni effettivi (calcolato tramite *fnContGG*) tra la data di inizio e il valore di *dataProd* originale. Ottenuto tale numero, effettua le stesse operazioni della funzione precedente, partendo dalla nuova data di inizio, per calcolare il nuovo valore di *dataProd*. Tra queste due date, infatti potrebbero esistere giorni non validi, ognuno dei quali farebbe ritardare di 24 ore la data di immissione in linea di un componente.

5.3 Elenco componenti visuali

Questa e le prossime sezione del presente capitolo sono dedicate alle funzioni ed ai controlli Visual Basic utilizzati nella realizzazione del modulo, partendo da un elenco di tutte le istanze di questi, suddivise per tipo, per poi entrare nel dettaglio solamente di quelle principali (tabella

A.8 in appendice), con particolare attenzione alla ridefinizione degli eventi scatenati dalle stesse (paragrafo 5.5).

`CommandButton`: il controllo CommandButton (pulsante di comando) viene utilizzato per iniziare, interrompere o concludere un processo. Quando viene scelto cambia aspetto e appare incassato.

Istanze create:
- cmdCentra
- cmdConn
- cmdGGNonValidi
- cmdLoad
- cmdResetScaler
- cmdSave
- cmdAllarga
- cmdRestringi
- cmdZoomIn
- cmdZoomOut

Form: l'oggetto Form è una finestra o una finestra di dialogo che fa parte dell'interfaccia utente dell'applicazione.

Istanze create:
- Form

Frame: il controllo Frame (cornice) consente di raggruppare controlli in modo visibile e identificabile. Può essere utilizzato per suddividere un form in modo funzionale, ad esempio separando gruppi di controlli OptionButton.

Istanze create:
- frmInfoLotto

- frmPrincipale

phGantX: questo controllo, appartenente all'OCX *phGantTimePackage*, consente di gestire risorse e dati legati al tempo tramite una visualizzazione grafica con cui è possibile interagire facilmente utilizzando il mouse.

Istanze create:
- Gantt

Label: Il controllo Label (etichetta) è un controllo di tipo grafico e consente di visualizzare del testo che l'utente può modificare direttamente nel caso in cui questa operazione sia stata abilitata in fase di programmazione.

Istanze create:
- lblColore
- lblFine
- lblGEffettivi
- lblGiorno
- lblInizio
- lblLinea
- lblMaga
- lblNumero
- lblOra
- lblStab

PictureBox: Il controllo PictureBox (casella immagine) consente di visualizzare un elemento grafico da file di diversi formati. Se le dimensioni del controllo non sono sufficientemente grandi per visualizzare l'intera immagine, l'elemento grafico verrà ritagliato.

Nel caso di utilizzo riguardante il progetto descritto nel presente documento, gli oggetti di questo tipo sono stati nascosti durante l'esecuzione del programma. Essi servono infatti esclusivamente come contenitori per le immagini raffiguranti lo stato della connessione.

Istanze create:
- picConnHolder
- picDisconnHolder

StatusBar: Un controllo StatusBar (barra di stato) visualizza una finestra, in genere nella parte inferiore di un form contenitore, che consente all'applicazione di visualizzare diversi tipi di dati relativi allo stato. Il controllo StatusBar può essere suddiviso in un massimo di sedici oggetti Panel contenuti in un insieme Panels.

Istenze create:
- sbStatusBar

WheelCatcher: un controllo di tipo WheelCatcher non è visibile durante una normale esecuzione del programma in cui viene incluso. Esso serve esclusivamente per catturare eventi scatenati dalla rotazione della rotellina del mouse. WheelCatcher non fa parte dei controlli standard di Visual Basic, deve pertanto essere aggiunto appositamente all'ambiente di sviluppo.

Istenze create:
- wcWheelCatcher

5.4 Funzioni in Visual Basic

Ho preferito includere alcune parti di codice, scritto in Visual Basic, in funzioni separate dagli eventi associati ai vari controlli, in modo da migliorarne la leggibilità e la comprensione ed evitare ripetizioni poco eleganti. La tabella A.9, presente in appendice, riassume e descrive le funzioni da me implementate indicandone il tipo di ritorno nel caso in cui queste ne abbiano uno (cioè se si tratta di Function, in alternativa si parla di Subroutine).

Nota: Per rendere più chiara la comprensione delle prossime sezioni del documento e le relative tabelle riportate in appendice, descrivo, qui di seguito, alcuni tipi di dati da me utilizzati nel corso dell'attività di stage.

Recordset: un oggetto recordset rappresenta il set di record ottenuto da una tabella di base o dai risultati di un comando eseguito. Questo tipo di oggetto fa riferimento solo ad un unico record all'interno del set come al record corrente. Gli oggetti recordset vengono utilizzati per manipolare i dati forniti da un provider. Tutti gli oggetti di tale tipo vengono creati utilizzando record (righe) e campi (colonne). A seconda della funzionalità supportata dal provider, è possibile che alcuni metodi o proprietà di un oggetto recordset non siano disponibili.

IphDataEntity_Tree2: questo tipo di dato, fornito dal controllo aggiunto phGantX, è paragonabile al nodo di una struttura ad albero. Partendo da un nodo radice (root) aggiunto all'istanza del controllo in questione tramite la funzione AddRootDataEntityTree invocata su tale istanza, è possibile costruire una struttura ad albero senza limiti di diramazione o livello. Nello specifico caso riguardante l'attività di stage da me svolta, questo tipo di dato è stato utilizzato per identificare e visualizzare lo stabilimento (considerato come radice dell'intera struttura dell'impianto del cliente), i magazzini (ovvero i

nodi di livello più alto appena sotto la radice) ed infine le linee di produzione (viste come foglie dell'albero, prive quindi di ulteriori nodi figli).

IphDataEntity_GantTime2: gli oggetti di questo tipo sono usati per rappresentare i blocchi di tempo all'interno dell'area grafica del controllo phGantX. Un oggetto IphDataEntity_GantTime2 detiene informazioni utili alla sua rappresentazione a video, come la data e l'ora di inizio e fine, il colore, lo stile, a quale riga è associato e molte altre. Nel caso relativo al modulo da me implementato, questo tipo di dato è stato utilizzato per rappresentare i lotti di produzione.

Segue un elenco di tutte le funzioni, da me aggiunte al codice scritto in Visual Basic, comprensivo delle liste di parametri in input. La tabella A.9 in appendice ne descrive invece il tipo, il tipo di ritorno ed il funzionamento.

- calcolaAltezzaRiga(recordset)
- connetti()
- contaCollisioni(long, recordset, long)
- disconnetti()
- getUserName()
- giornoValido(date, string, string, string)
- initActivity(IphDataEntity_Tree2, recordset)
- initTimeItem(IphDataEntity_GantTime2, recordset)
- loadTimeItemsOwnedBy(IphDataEntity_Tree2, variant)
- loadTreeItemsOwnedBy(IphDataEntity_Tree2, variant)

5.5 Eventi principali

In questa sezione verranno descritti gli eventi, funzionalmente più importanti, scatenati dai controlli principali del modulo implementato.

Controllo: **cmdConn**

Eventi ridefiniti:

- **Click**: quando il cursore si trova sull'area occupata da questo controllo, se viene premuto il pulsante sinistro del mouse viene richiamata la funzione *connetti* nel caso in cui non esista una connessione tra il modulo ed il database, in caso alternativo viene richiamata *disconnetti*. A seconda dello stato della connessione, il controllo assume l'immagine più rappresentativa tra quella detenuta da *picConnHolder* e *picDisconnHolder*.

Controllo: **cmdLoad**

Eventi ridefiniti:

- **Click**: quando il cursore si trova sull'area occupata da questo controllo, se viene premuto il pulsante sinistro del mouse viene inizialmente eseguita una query al database in modo da verificare l'esistenza di dati nella tabella temporanea, che facciano riferimento all'utente che sta utilizzando il modulo. Nel caso in cui la query ritorni una risposta affermativa, viene visualizzata una finestra pop-up che permetta all'utente di ricaricare la situazione precedentemente elaborata ma mai salvata oppure di mostrare la situazione che rappresenta lo stato attuale dei dati attualmente salvati nel database. Se l'utente effettuasse questa seconda scelta o nel caso in cui la query ritorni una risposta negativa, la tabella temporanea viene sovrascritta con i nuovi dati riguardanti i lotti attualmente accodati tra le linee produttive e, ad ognuno di tali lotti, viene assegnato un colore diverso in modo da essere immediatamente distinguibile dagli altri all'occhio dell'utente.

Controllo: **cmdSave**

Eventi ridefiniti:

- **Click**: quando il cursore si trova sull'area occupata da questo controllo, se viene premuto il pulsante sinistro del mouse appare una finestra pop-up che, semplicemente, richiede una conferma da parte dell'utente riguardo l'operazione scelta. Questo passaggio ha il solo scopo di evitare modifiche involontarie al database, ad esempio, nel caso in cui il pulsante descritto venga premuto accidentalmente. Una volta confermata l'azione, viene eseguita una query alla tabella temporanea che, tramite il controllo del campo *modificato*, ritorna tutti i lotti movimentati dall'utente. A questo punto, per ogni lotto trovato, viene eseguita un'ulteriore query, questa volta di update, che aggiorna il campo *dataProd* di ogni record di Ordine_Dettaglio che faccia riferimento ad una delle commesse appartenenti al lotto corrente. Il nuovo valore di *dataProd*, come precedentemente detto, viene scelto tramite la funzione *calcolaNuovaDataProd*. Per ognuno di tali lotti viene creato un record nella tabella Log_Gestione_lotti che permetta di risalire allo stato precedente del database, nel caso in cui l'utente abbia effettuato modifiche errate al piano principale di produzione.
  Se, durante il ciclo di update, la connessione al database dovesse accidentalmente interrompersi o si verificasse un errore qualsiasi, le operazioni eseguite fino a quel punto verrebbero annullate, in modo da mantenere coerenza tra i dati.

Controllo: **Form**

Eventi ridefiniti:

- **Load**: il lancio dell'applicazione scatena l'evento di caricamento del *form* principale. Questa fase di attivazione si occupa di impostare le

caratteristiche principali del controllo di tipo phGantX contenuto nella finestra e di disabilitare gli opportuni pulsanti. Inizialmente, infatti, l'interfaccia non è connessa al database e non è quindi possibile effettuare il caricamento dei dati o salvare la situazione visualizzata. Non ha senso, in questo caso, che i controlli responsabili di tali operazioni, siano disponibili.

- **MouseMove**: quando il cursore del mouse si sposta in una posizione qualsiasi all'interno del Form, che non sia occupata dal controllo di tipo phGantX, le infomrazioni eventualmente visualizzate nelle label *lblGiorno*, *lblOra* e in tutte quelle raggruppate nel frame *frmInfoLotto*, vengono cancellate.

- **Resize**: qualora l'utente decida di modificare le dimensioni di default della finestra principale, l'evento *Resize* ha il compito di mantenere le proporzioni tra i controlli contenuti nel *Form* garantendo, per qualunque combinazione di altezza e larghezza di quest'ultimo, la visualizzazione più gradevole e funzionale.

- **Unload**: tramite l'invocazione della funzione *disconnetti*, questo evento determina la terminazione della connessione tra il modulo ed il database. Esso viene scatenato quando l'utente chiude la finestra dell'applicazione tramite l'opportuno pulsante.

Controllo: **Gantt**

Eventi ridefiniti:

- **OnGantAreaDrawBackground**: questo evento viene scatenato nel momento in cui il sistema operativo della macchina sulla quale si sta eseguendo il modulo, disegna l'area grafica sullo schermo (operazione effettuata a sua volta dall'evento *Load* del controllo *Form*). Il suo unico scopo è quello di dare un riferimento visivo

all'utente in merito al giorno ed all'ora attuali, riportandoli direttamente sul grafico temporale sotto forma di una spessa linea nera perfettamente verticale.

- **OnHintInfo**: appena il puntatore del mouse entra nell'area contenuta in uno degli oggetti rettangolari rappresentanti i lotti di produzione, tutti i controlli di tipo label, raggruppati nel frame *frmInfoLotto*, vengono inizializzati con le opportune informazioni. Tramite questi si possono infatti leggere il numero seriale del lotto puntato, la data di inizio, quella di fine, il numero di giorni effettivamente occupati, lo stabilimento, il magazzino e la linea a cui è associato e, per finire, il colore con cui è disegnato, in modo da disambiguare un'eventuale posizione poco chiara del cursore tra più oggetti ravvicinati.

- **OnMouseMove**: gli spostamenti orizzontali del puntatore del mouse, all'interno dell'area occupata dal grafico del controllo *Gantt*, modificano il contenuto delle label *lblGiorno* ed *lblOra*, le quali, in ogni istante, riportano rispettivamente la data e l'ora corrispondenti al pixel puntato dal cursore nel caso in cui questo si trovi nella suddetta area. Se, inoltre, il puntatore esce dall'area contenuta in uno degli oggetti rettangolari rappresentanti i lotti di produzione, questo evento si occupa di cancellare il contenuto di tutte le label raggruppate in *frmInfoLotto*.

- **OnSelectionChangedGrid**: questo evento, di importanza marginale, è responsabile unicamente della modifica delle caratteristiche visive dell'eventuale riga del controllo grafico attualmente selezionata.

- **OnValueChangedGantTime**: ogni volta che un oggetto rettangolare, corrispondente ad un lotto produttivo, subisce uno spostamento che ne modifichi le date di inizio e fine montaggio, questo evento viene attivato. Per prima cosa controlla, tramite l'invocazione della funzione *giornoValido*, che il giorno scelto come

nuovo inizio sia valido. Nel caso in cui tale funzione ritorni una risposta negativa, viene preso in esame il giorno successivo e così via fino a trovare una data accettabile. Scelta la data di inizio, viene nuovamente utilizzata *giornoValido* per trovare la nuova data di fine. A questo punto è possibile, tramite *calcolaAltezzaRiga* ricalcolare il numero massimo di collisioni (sovrapposizioni) tra i lotti associati alla linea di quello appena spostato in modo da poter definire la nuova altezza da assegnare alla riga del grafico considerata. Infine, il corpo dell'evento si occupa di aggiornare il record di TMP_Gestione_Lotti impostando ad 1 il campo m*odificato* e sovrascrivendo i campi *nuovoIni* e *nuovaFine* con i dati scelti dall'utente.

Capitolo 6

Test finali, documentazione e scelte effettuate

6.1 Test di soddisfacimento dei requisiti

A partire dai requisiti individuati, sia nella fase di sviluppo che in quella di collaudo, è stata eseguita una serie di test al fine di verificare che il modulo rispondesse alle richieste del cliente.

Oltre alle prove effettuate tramite l'interfaccia, una volta iniziata la fase di collaudo, ogni componente o funzione contenente parti di codice per dialogare con il database, è stata testata durante il proprio sviluppo. Per fare ciò, le query di selezione, cancellazione o aggiornamento, utilizzate in tali sezioni, sono state provate da riga di comando tramite SQL Server 2005. Una volta appurata la correttezza del risultato prodotto, le query sono state importate nel codice Visual Basic incaricato di richiamarle.

La tabella riportata di seguito indica i test effettuati sul modulo implementato seguendo l'elenco dei requisiti da soddisfare. La colonna più a destra contiene i risultati prodotti dai test una volta apportate le dovute correzioni. Per motivi di spazio viene fatto riferimento solamente alle sigle dei requisiti (descritti per esteso nell'elenco del terzo capitolo di questo documento).

Requisito	Test effettuato	Risultato ottenuto
0.F.s	Movimentazione lotti tramite la nuova interfaccia e verifica della correttezza dei cambiamenti tra i record del database coinvolti	Effettuando lo spostamento, sull'asse temporale, prima di un singolo lotto e poi di più lotti, dopo il salvataggio, tramite l'apposito pulsante, i dati del database, relativi ai lotti considerati, sono stati correttamente aggiornati con le date di inizio e fine scelte.

Requisito	Test effettuato	Risultato ottenuto
1.F.s	Movimentazione lotti tramite la nuova interfaccia e verifica della correttezza dei cambiamenti tra i record del database coinvolti	Effettuando lo spostamento, sull'asse temporale, di un lotto l'unico campo aggiornato, una volta premuto il pulsante di salvataggio, è *dataProd*, relativo ai record di Ordini_Dettaglio che fanno riferimento alle commesse del lotto considerato.
2.F.s	Movimentazione lotti, presenti su più linee di produzione, tramite la nuova interfaccia e verifica della correttezza dei cambiamenti tra i record del database coinvolti	Dato un lotto, le cui commesse sono distribuite su due o più linee, effettuando lo spostamento della porzione di questo, limitatamente ad una sola riga del grafico, le date di inizio e fine delle altre parti dello stesso, non vengono influenzate.
3.F.s	Movimentazione lotti, riavvio del modulo e ripristino dati da tabella temporanea	Dopo aver modificato il piano principale di produzione, tramite l'interfaccia, effettuando la disconnessione dal database o interrompendo l'esecuzione del programma senza effettuarne il salvataggio, una volta riavviato il modulo e caricati i dati presenti nella tabella temporanea, la situazione precedentemente impostata viene correttamente ripristinata.
4.F.s	Improvvisa disconnessione dal database o imprevisto arresto del modulo	Grazie all'utilizzo di opportune transazioni, rendendo atomiche le operazioni più articolate sul database, nel caso in cui si verifichi un errore, che non dipenda dall'utilizzo sbagliato del modulo, durante una di queste, l'integrità dei dati non viene compromessa.
5.F.s	Spostamento di un lotto su un intervallo di tempo contenente una o più date non valide per la linea presso cui questo è accodato	Posizionando un lotto, di lunghezza non trascurabile, modificandone le date di inizio e fine in modo che, tra queste, vi sia almeno un giorno contenuto nella tabella Giorni_Non_Lavorativi, associato alla linea di produzione riguardante il lotto considerato, sia la visualizzazione di questo che l'aggiornamento delle date di montaggio dei componenti, vengono opportunamente modificate.

Requisito	Test effettuato	Risultato ottenuto
6.Q.s	Salvataggio della nuova situazione visualizzata	Ad ogni pressione del tasto di salvataggio, per ogni lotto movimentato viene creato un record nella tabella Log_Gestione_Lotti, nel quale vengono registrate tutte le informazioni utili all'eventuale ripristino del precedente stato del database, con indicazioni relative all'utente responsabile dell'operazione.
7.Q.s	Simulazione crash di sistema o imprevista interruzione della connessione di rete	Grazie al dettagliato ed efficace modulo di gestione delle eccezioni creato dagli sviluppatori di Automa, da me importato nel codice in Visual Basic del modulo di gestione del piano principale di produzione, ogni errore dovuto ad un guasto hardware o di altra natura, viene opportunamente registrato in un file di log.
8.Q.s	Simulazione crash di sistema o imprevista interruzione della connessione di rete	Dal momento che ad ogni spostamento di un singolo lotto tramite l'interfaccia da me sviluppata, la tabella temporanea viene opportunamente aggiornata, qualunque errore improvviso non influenza il lavoro svolto dall'utente fino a quel momento.
9.Q.s	Spostamento di un lotto su un intervallo di tempo contenente uno o più giorni festivi o appartenenti al fine settimana	Dal momento che il sabato e la domenica non vengono visualizzati nel grafico temporale, la lunghezza della settimana è considerabile di cinque giorni. Per quanto riguarda le date festive, in cui tutto lo stabilimento rimane chiuso, queste vengono trattate, come per il requisito 5.F.s, relativamente ad ogni linea di produzione dell'intero impianto.
10.Q.u	Pressione del tasto di caricamento dati a connessione effettuata	Premendo il pulsante responsabile del caricamento e della successiva visualizzazione dei dati, nel caso in cui, nella tabella temporanea, siano presenti record associati all'utente che stia attualmente utilizzando il modulo, viene proposto, a questo, di caricare la situazione precedentemente impostata ma non ancora salvata o visualizzare il piano principale di produzione senza considerare le modifiche apportate.

Requisito	Test effettuato	Risultato ottenuto
11.Q.u	Pressione del tasto di salvataggio	Dopo aver movimentato uno o più lotti, premendo il pulsante di salvataggio, una finestra di pop-up segnala all'utente la delicatezza dell'operazione scelta, permettendo a questo di annullarla o renderla effettiva.
12.Q.u	Pressione del tasto di connessione/ disconnessione in ogni momento in cui sia possibile farlo	Premendo il pulsante incaricato della gestione della connessione del modulo al database, questa viene opportunamente interrotta o ristabilita a seconda dello stato attuale del programma.
13.Q.u	Ridimensionamento della finestra	Premendo il pulsante di estensione a schermo intero del form o trascinando i bordi della finestra, gli elementi all'interno di questa mantengono le proporzioni permettendone una buona visualizzazione per qualsiasi combinazione di altezza e larghezza.
14.Q.u	Esecuzione di operazioni che impieghino un lasso di tempo percepibile dall'utente	Durante l'esecuzione di operazioni che tengano impegnato il modulo per un intervallo di tempo relativamente lungo, la barra di stato del form avvisa l'utente di cosa stia accadendo all'interno del sistema. La temporanea modifica dell'aspetto del puntatore del mouse aiuta ulteriormente l'utente a rendersi conto che il modulo non è bloccato a causa di errori.
15.Q.u	Posizionamento del puntatore del mouse su un qualsiasi lotto di produzione	Non appena il cursore del mouse entra nell'area occupata da un lotto visibile nell'area grafica, i campi contenuti nel riquadro *Informazioni lotto focalizzato*, vengono riempiti con tutti i dati importanti relativi all'elemento considerato.
16.Q.u	Esecuzione di ogni operazione possibile dal modulo	Considerando il fatto che i test sono stati effettuati dagli uffici di Madlab (a Padova), connettendo il modulo al server di Manitou (di Modena), e che, una volta installato il componente, questo lavorerebbe in locale, i tempi di attesa, anche per le operazioni più onerose, si sono dimostrati accettabili.

Requisito	Test effettuato	Risultato ottenuto
17.Q.u	Caricamento dati ripetuto più volte	Ad ogni caricamento dati che non consideri i record contenuti nella tabella temporanea, un nuovo set di colori viene assegnato ai lotti di produzione evitando ripetizioni (e quindi ambiguità) pur mantenendo la stessa tonalità per lo stesso lotto eventualmente distribuito su più linee.
18.Q.u	Pressione del tasto *Reimposta visualizzazione* o *Trova momento attuale* in seguito a cambiamenti della visualizzazione del grafico	Premendo gli opportuni tasti, dopo essersi spostati sull'asse temporale, anche fino agli estremi di questo, o dopo aver modificato la dimensione dell'intervallo visualizzato, è possibile tornare facilmente alle impostazioni di default del grafico.
19.Q.u	Pressione dei tasti *Allarga/Restringi visuale linea*	Selezionando una riga qualsiasi del grafico e premendo gli opportuni pulsanti, l'altezza di questa può essere modificata a piacimento.
20.Q.u	Pressione dei tasti *Zoom In* e *Zoom Out*	Premendo gli opportuni pulsanti, in qualsiasi momento, durante l'esecuzione del programma, le dimensioni dell'intervallo di tempo visualizzato vengono correttamente modificate
21.Q.u	Esecuzione del modulo	La dimensione dei pulsanti è stata pensata per utenti che abbiano poca dimestichezza con dispositivi di puntamento (tipo mouse o touchpad). E' inoltre possibile spostarsi tra questi utilizzando esclusivamente la tastiera. Posizionando il puntatore del mouse su un pulsante qualsiasi, un'etichetta a comparsa aiuta la comprensione della funzione di quest'ultimo.

6.2 Documentazione

Gli sviluppatori di Automa non hanno dedicato tempo alla documentazione riguardante l'analisi, la progettazione, l'implementazione, i test e la manutenzione del software per quanto complesso ed esteso sia diventato negli anni. Coerentemente a questa scelta (a mio avviso

discutibile), non mi è stata rivolta alcuna richiesta in merito a documenti sul lavoro svolto all'interno dell'azienda. Per sopperire a tale mancanza e spinto dal timore di non ricordare alcuni dettagli importanti, ho dedicato alcune ore, al di fuori di quelle previste per l'attività di stage, al fine di produrre documentazione (informale) il cui contenuto è riportato all'interno della presente tesi. Ho, inoltre, provveduto alla stesura di un breve manuale utente ed una guida multimediale basata su un esempio di utilizzo del modulo.

6.3 Scelte implementative

Nonostante, il più delle volte, le strade da seguire nell'implementazione del modulo fossero abbastanza forzate, a mio avviso vale la pena fare qualche esempio riguardante alcune mie scelte di sviluppo per le quali ho avuto più libertà.

Uno dei problemi marginali che ho dovuto affrontare, riguardava la colorazione degli oggetti rappresentanti i lotti. Dopo aver provato ad utilizzare la gamma di colori messa a disposizione da Visual Basic, assegnando una diversa tonalità ad ogni lotto al momento della rappresentazione a video della situazione del database, ho deciso di operare direttamente sulla tabella temporanea. Per fare ciò, dopo aver trovato in Internet un elenco dei valori esadecimali corrispondenti ad un soddisfacente numero di colori, ho creato una tabella (chiamata *Colori_Hex*) con una sola colonna e l'ho popolata con tale elenco. Ho poi modificato la tabella *TMP_Gestione_Lotti* aggiungendo la colonna *colore*. Ogni volta che i record di *TMP_Gestione_Lotti* vengono sovrascritti dall'evento *Click* del controllo *cmdLoad*, con l'ausilio della funzione di T-SQL, *newid*, usata per sorteggiare casualmente i valori di *Colori_Hex*, assegno un nuovo set di colori ai lotti esistenti, evitando ripetizioni ad eccezione degli oggetti appartenenti a righe diverse ma rappresentanti lo stesso lotto.

Un altro problema riscontrato, anch'esso legato ad un requisito più di tipo qualitativo che funzionale, riguardava il calcolo dell'altezza della riga del grafico, più adeguata alla situazione in essa rappresentata. Spesso, infatti, gli intervalli di tempo tra l'inizio e la fine di diversi lotti associati alla stessa linea, si accavallano tra loro. Nella pratica, tale situazione si verifica quando nella coda di commesse appartenenti ad un certo lotto ne vengono inserite altre che non abbiano nulla a che fare con questo.

Il controllo di tipo phGantX da me utilizzato fornisce la possibilità di ridefinire un evento, chiamato *OnCollisionDetect*, scatenato nel caso in cui si verifichi la sovrapposizione tra due elementi adagiati sulla medesima riga o, viceversa, ogni volta che viene annullato l'accavallamento tra due lotti precedentemente sovrapposti. Il problema è che gli sviluppatori del controllo non hanno messo a disposizione dei programmatori la possibilità di discriminare tra questi due casi. Per tale motivo ho dovuto fare a meno di tale evento. L'alternativa a cui ho pensato è relativamente semplice. L'insieme dei lotti associati ad una linea (visualizzati quindi in un'unica riga) sono ordinati per data di fine crescente. Tale insieme viene passato alla funzione *calcolaAltezzaRiga* che, per ogni lotto preso nell'ordine dato, invoca *contaCollisioni*. Quest'ultima, tramite un controllo iterativo, incrementa un contatore di un numero intero ogni volta che trova un lotto in posizione successiva nell'ordine dato ma con data di inizio minore della data di fine di quello passato. Il più alto numero di collisioni ritornato da *contaCollisioni* è il parametro su cui si basa *calcolaAltezzaRiga* per determinare l'altezza più adatta per la riga considerata.

Tutto ciò viene fatto al caricamento dei dati per ogni riga oppure in seguito allo spostamento di un lotto per mano dell'utente, in quest'ultimo caso limitatamente alla riga su cui il lotto è visualizzato.

Il terzo e ultimo problema che voglio descrivere riguarda la scelta della struttura temporanea in cui salvare la situazione elaborata dall'utente prima che questa venga riportata sul database sovrascrivendone i dati. Inizialmente, una volta appresa l'utilità e la praticità degli oggetti di tipo recordset, la strada da me scelta è stata quella di costruire una struttura

ricorsiva (ad albero) completamente basata su questi. In poche parole, un recordset radice, associato allo stabilimento, sarebbe stato composto dai campi riguardanti le informazioni relative a quest'ultimo e da un ulteriore campo, anch'esso di tipo recordset, contenente i recordset relativi ai magazzini. Questi, a loro volta, oltre ai campi per le informazioni, ne avrebbero avuto un altro, di nuovo di tipo recordset, che sarebbe servito per la memorizzazione delle linee. Lo stesso ragionamento sarebbe stato applicato all'ultimo livello, in cui ogni record riguardante una linea, avrebbe contenuto un recordset con lo scopo di mantenere tutti i lotti accodati presso di essa.

Nonostante la mia idea fosse stilisticamente elegante e funzionalmente efficiente, mi è stato fatto notare che, dati i tempi ristretti, sarebbe stato meglio optare per una soluzione più semplice. Appurata, infatti, la complessità della mia strategia, ho ripiegato sulla tabella temporanea, il cui funzionamento è stato descritto precedentemente.

6.4 Esempio di utilizzo e descrizione interfaccia

Per rendere più chiara la comprensione di quanto esposto fin qui e dare un'idea di come si presenti e funzioni il modulo per la gestione del piano principale di produzione, in questa sezione del documento sarà descritto il comportamento del programma con l'aiuto di immagini esemplificative.

La figura 6.1 ritrae l'interfaccia grafica al momento dell'avvio del programma. Nessun dato è visualizzato e le uniche operazioni permesse riguardano la connessione al database, lo spostamento del grafico sull'asse del tempo o la modifica dell'intervallo visualizzato (che di default è di 10 giorni).

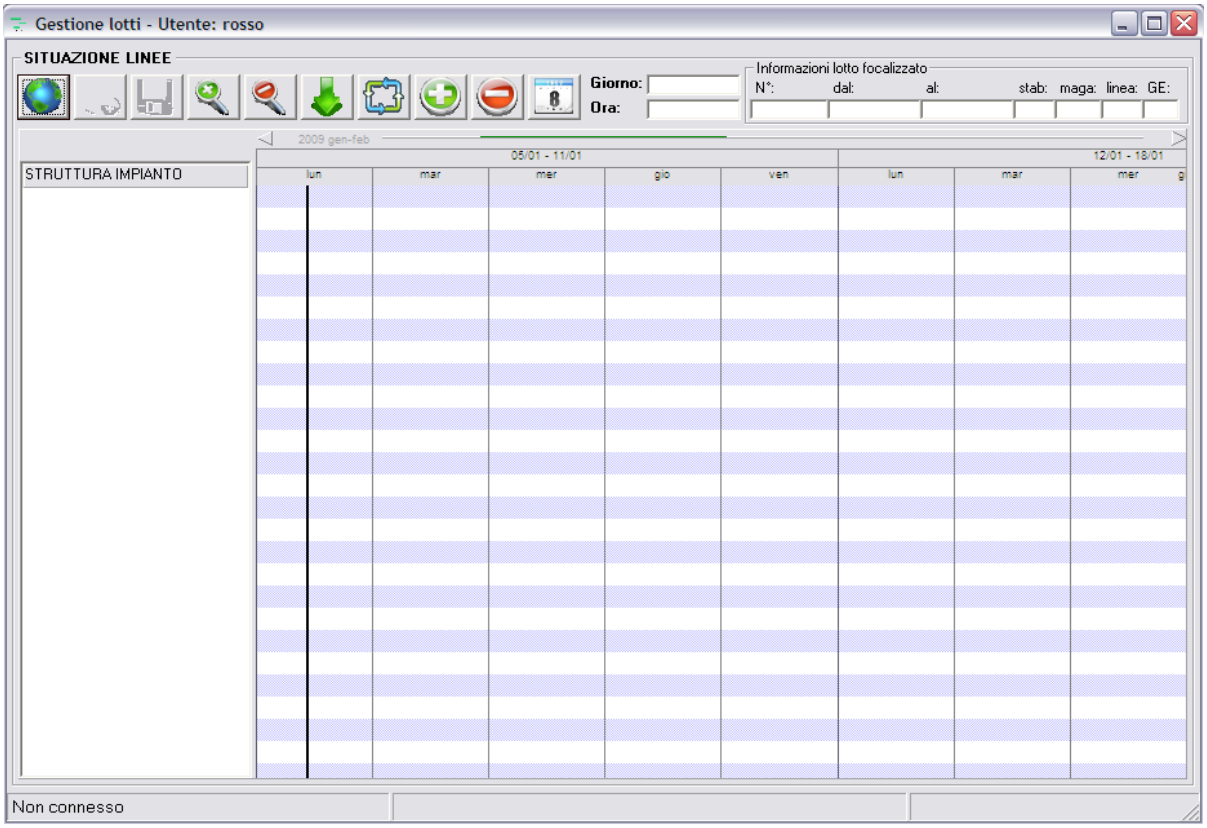

Figura 6.1 - Esempio interfaccia all'avvio del programma

Una volta premuto il pulsante di connessione (primo a sinistra nella figura 6.2), viene aggiunta, tra le azioni appena descritte, la possibilità di caricare i dati dal database tramite il comando associato al secondo pulsante (abilitato una volta stabilito il collegamento al database).

Figura 6.2 - Dettaglio pulsantiera

Dopo aver scelto se visualizzare i dati contenuti nella tabella temporanea o caricare i lotti realmente accodati nel momento attuale presso le linee di produzione, attendendo qualche secondo, l'interfaccia assume un aspetto simile all'esempio riportato in figura 6.3.

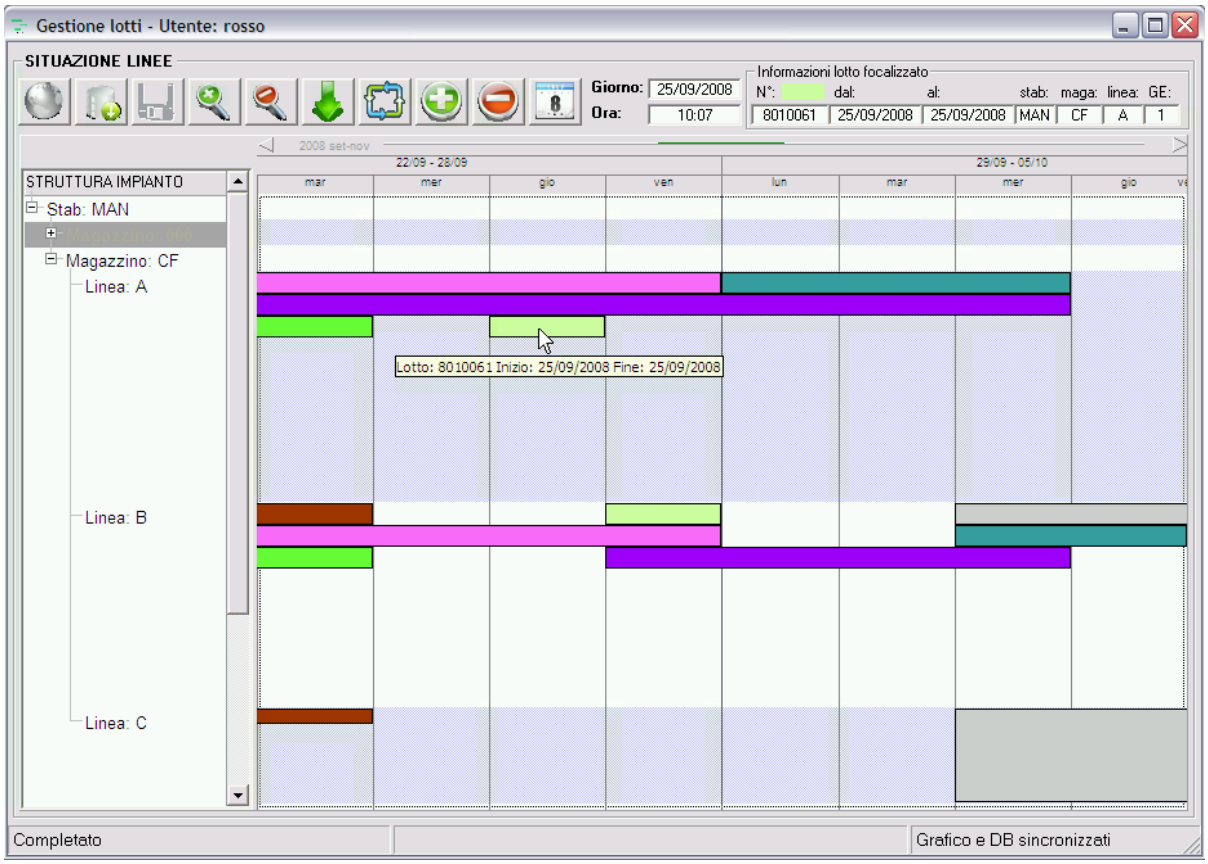

Figura 6.3 - Esempio utilizzo post caricamento dati

A questo punto, la colonna più stretta, a sinistra della finestra, mostra la struttura dell'impianto del cliente tramite una visualizzazione ad albero in cui la radice rappresenta lo stabilimento, i nodi di primo livello raffigurano i magazzini mentre le foglie le linee di produzione.

L'area più estesa della finestra (con sfondo a colori alternati per evitare ambiguità tra le linee), a destra della suddetta colonna, contiene ora gli elementi grafici rappresentanti i lotti e le righe di questo riquadro sono associate ai nodi dell'albero di sinistra. Ogni riga in corrispondenza di una foglie dell'albero, e cioè di una linea di produzione, raggruppa tutti i lotti accodati presso tale linea.

Tramite il solo uso del mouse è ora possibile trascinare i lotti modificandone le date di inizio e fine produzione, fino a raggiungere la

situazione desiderata.

Quando il puntatore del mouse si trova sopra un lotto, vengono visualizzate le caratteristiche di questo tramite i campi contenuti nel riquadro *Informazioni lotto focalizzato* (figura 6.4). A sinistra di tale riquadro, altri due campi riportano la data e l'ora corrispondente all'esatta posizione del puntatore del mouse nel caso in cui questo si trovi all'interno dell'area occupata dal grafico temporale.

Figura 6.4 - Dettaglio campi contenenti informazioni lotto focalizzato e coordinate mouse in termini di data e ora

A partire dalla situazione caricata direttamente dal database (ovvero senza considerare il contenuto della tabella temporanea), dopo il primo spostamento di un lotto qualsiasi tra tutti quelli visualizzati, viene abilitato il pulsante di salvataggio (il terzo da sinistra in figura 6.2), tramite il quale è possibile aggiornare il database rendendo effettivo il nuovo piano principale di produzione scelto dall'utente.

Capitolo 7

Valutazioni e conclusioni finali

7.1 Tempistiche

Come si può notare dal grafico riportato qui di seguito, i tempi previsti non sono stati perfettamente rispettati ad eccezione della fase di analisi.

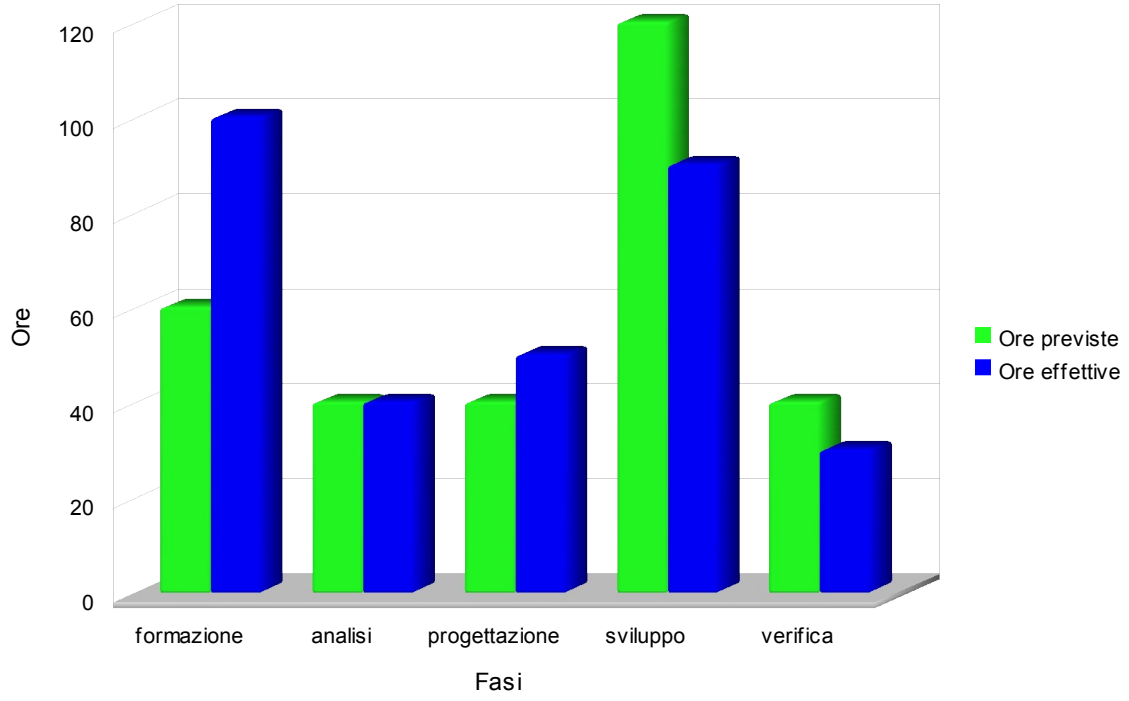

Figura 7.1 - Grafico di confronto tra tempi previsti ed effettivi

Le cause dei principali imprevisti sono state la mancanza di documentazione e di commenti nel codice di Automa, la complessità dello stesso, la scarsa qualità della documentazione relativa all'OCX aggiunto ai controlli di Visual Basic ed il fatto che nessuno tra il personale di Madlab

avesse mai avuto a che fare con questo componente alquanto complicato da utilizzare. La stretta relazione tra la logica adottata da Manitou, la complessità della struttura operativa di questo cliente ed il lavoro che ho dovuto svolgere, hanno prolungato ulteriormente la fase di formazione iniziale.

In compenso, grazie al dettagliato studio del problema, la fase di sviluppo si è rivelata più breve del previsto, facendomi quasi rientrare nelle trecento ore prestabilite per l'intera attività di stage.

Dal momento che, nel piano di lavoro presentato ad inizio stage, le due settimane assegnate alla stesura della documentazione eccedevano i limiti di tempo imposti dal corso di studio, ho preferito non riportarle nel grafico, considerando anche il fatto che i documenti sono stati prodotti sfruttando ore straordinarie rispetto a quelle di regolare attività in azienda.

7.2 Prossimi sviluppi

Inizialmente il periodo di stage previsto avrebbe dovuto avere una durata doppia rispetto a quella imposta dal regolamento del corso di studi. Lo scopo a cui Madlab intendeva farmi arrivare, si spingeva oltre a ciò che sono riuscito a realizzare. Il dimezzamento dei tempi, ci ha costretti a spezzare il lavoro, rimandando la seconda parte ad un'ulteriore stage post-laurea o una volta steso un contratto lavorativo tra me e l'azienda.

Di seguito viene esposta, in modo conciso, l'idea su cui si basa l'estensione prevista per il modulo da me creato durante questi primi mesi di rapporto con Madlab. E' utile però, a mio avviso, approfondire prima alcuni aspetti relativi alla logica con cui Automa opera, per rendere più chiaro il concetto.

7.2.1 Logica adottata

Per sapere in ogni istante il numero di pezzi disponibili di ogni articolo e dove sono situati, Madlab utilizza il concetto di “missione”. Qualunque

azione venga compiuta all'interno dello stabilimento è guidata da una missione la quale può generare a sua volta altre missioni. E' utile, a mio avviso, cercare di pensare a queste missioni come a singoli thread in concorrenza tra loro per l'acquisizione delle risorse in comune, corrispondenti agli articoli del magazzino, ognuna con molteplicità pari alla giacenza del rispettivo articolo in tutto lo stabilimento.

Nel caso specifico di Manitou, viene generato un ordine di tante commesse quante sono le macchine ordinate da un certo cliente. Ogni commessa (macchina finita che verrà consegnata) è vista dal sistema come un insieme di ordini di dettaglio che fanno tutti capo allo stesso ordine di testata (record che lega quella particolare commessa al cliente che l'ha ordinata e dice quale linea di produzione deve prendersi carico dell'assemblaggio della stessa).

Ogni ordine di dettaglio si riferisce ad un certo kit (componente) della macchina che si desidera produrre. Una macchina infatti è concepita in Manitou come un oggetto modulare, composto da altri oggetti (kit) che sono comuni tra le diverse macchine, a seconda delle esigenze del cliente. In definitiva, il prodotto finito è una combinazione di kit, alcuni dei quali si escludono tra loro vicendevolmente.

Un kit fa riferimento a sua volta a *n* righe d'ordine, ognuna delle quali descrive quanti pezzi di un determinato articolo servono per quel particolare kit e in che postazione devono essere portati per poi essere assemblati.

A questo punto entrano in gioco le missioni; ogni riga d'ordine infatti, da vita a:

- una missione nel caso in cui il numero di pezzi dell'articolo descritto dalla riga siano trasportabili “atomicamente” all'unica postazione che necessita dell'articolo considerato (ad esempio perché il prelievo coinvolge una singola Udc o il “pallet intero”);
- più missioni se le postazioni che necessitano di quel tipo di articolo sono più di una o se i pezzi di quel medesimo articolo sono dislocati in diverse celle del magazzino o in diverse aree dello stabilimento. In quest'ultimo caso infatti, è necessario creare un numero sufficiente di missioni relative allo stesso articolo tale per cui, una

volta raggruppate tutte queste, si abbia un numero di pezzi maggiore o uguale a quello richiesto dalla rispettiva riga d'ordine.

In realtà una missione non è altro che un record di una tabella del database tramite la quale vengono associate tra loro la riga d'ordine che l'ha generata, l'articolo desiderato, la quantità richiesta, la locazione dei pezzi da movimentare più comoda per l'operazione che ne ha richiesto il prelievo, la locazione in cui i pezzi devono essere trasportati, il magazziniere che deve effettuare lo spostamento, altri dati trascurabili in questo contesto.

Dal momento che, come già detto in precedenza, ogni movimento interno allo stabilimento viene tracciato nel database tramite record di tipo missione, queste sono state catalogate tenendo conto della loro diversa natura. Esistono infatti:

- *missioni di prelievo*: generate da un ordine di immissione in linea di un lotto di produzione;
- *missioni di versamento*: generate da un magazziniere (attraverso l'apposito terminale ottico a radio frequenza) durante il prelievo di un insieme di diversi tipi di materiale o di pezzi dello stesso articolo dislocati in diverse aree;
- *missioni di avanzamento*: generate da ogni postazione in seguito ad un avanzamento della linea di produzione (programmato per scattare ogni *n* minuti). Effettuano essenzialmente lo "scarico" degli articoli montati in quella determinata postazione in modo da non sovraccaricare il sistema. E' infatti inutile continuare a tenere traccia dei movimenti di un articolo una volta che questo è stato montato per dare vita al prodotto finale;
- altre missioni la cui funzione esula dal contesto trattato.

Grazie a tali record, Automa riesce a gestire la disponibilità di ogni articolo sommando le quantità di pezzi dello stesso, impegnate da varie missioni in seguito all'ordine di una o più commesse. Tale funzionalità è indispensabile al fine di sapere se, al momento del lancio in linea di un lotto di produzione, le giacenze presenti nello stabilimento sono sufficienti per portare a termine gli assemblaggi entro i tempi desiderati, segnalando eventualmente di quali articoli e in che quantità è necessario rifornirsi al

più presto. Questa operazione viene eseguita in Automa con il nome di "simulazione" (termine improprio in quanto, una volta lanciata, nel caso in cui le giacenze non siano sufficienti, elimina le righe delle missioni "simulate" finora, in alternativa rende effettive tutte le missioni necessarie alla produzione del lotto senza chiedere conferma all'utente).

7.2.2 Lo scopo finale

Nonostante il lavoro da svolgere per aggiungere la funzionalità pensata da Madlab sia davvero poco banale, è descrivibile in modo semplice e sintetico.

Ciò che si desidera offrire al cliente è la possibilità di manovrare la produzione in modo molto più approfondito rispetto alle potenzialità fornite dal modulo implementato durante lo stage. In pratica, una volta impostato il nuovo piano principale di produzione, come descritto in precedenza, l'estensione deve permettere all'utente di prevedere le conseguenze delle proprie scelte (in termini di giacenze, numero di mancanti, forniture necessarie, ecc.) lanciando un processo che, tramite le suddette "missioni", simuli la situazione finale nel caso in cui venisse applicata la strategia di produzione visualizzata in quel momento nell'interfaccia di gestione. Le risposte ritornate da un procedimento simile sarebbero molto utili per uno studio più accurato da parte dell'utente sulle decisioni da prendere.

7.3 Conclusioni

L'attività di stage si è svolta in modo soddisfacente nonostante gli imprevisti e le difficoltà incontrate soprattutto nelle fasi di apprendimento e sviluppo. Tutti gli obbiettivi proposti prima di iniziare il lavoro sono stati raggiunti ed il prodotto finale è già utilizzabile. Tuttavia, ad oggi, non è stato possibile installare il modulo presso il sistema del cliente per diversi motivi.

L'ambiente di lavoro in cui mi sono trovato ad operare si è rivelato fin da subito accogliente, amichevole ed informale ma non per questo poco professionale. Il tutor aziendale, assieme a tutto il resto del personale di Madlab, si è dimostrato disponibilissimo, gentile, paziente e molto competente. Si è inoltre detto soddisfatto di come ho svolto il compito assegnatomi e, come accennato ad inizio stage, ha affermato di essere ancora convinto in merito all'assunzione del sottoscritto, una volta conseguita la laurea.

Pur sapendo che lo stage che avrei dovuto affrontare, per completare il corso di laurea, sarebbe stata un'esperienza utile per mettere in pratica ciò che ho imparato durante gli studi, non avrei mai pensato che mi potesse insegnare tanto. Oltre ad imparare un nuovo linguaggio di programmazione ed ad utilizzare strumenti con cui non avevo mai interagito prima, ho potuto iniziare ad avere un rapporto diretto con il cliente ed immergermi, anche se per un periodo di tempo limitato, nella realtà lavorativa che dovrò affrontare in futuro.

Appendice

A.1 Ordini_Testata

Campo	Tipo	Obbligatorio	Descrizione
stab	varchar	si	Indica lo stabilimento dal quale l'ordine verrà evaso. Per tutti i clienti di Manitou (o quasi) esiste un solo stabilimento, per cui questo campo è spesso non indispensabile. Viene coinvolto nel gruppo di campi che definiscono la chiave primaria di praticamente tutte le tabelle del database, come in questo caso. Per tale motivo l'attributo non può mai assumere valore *null*.
maga	varchar	si	Un cliente può avere più di un magazzino. E' il caso proprio di Manitou che ne possiede due a Modena. Questo campo suddivide i dati tra quelli che fanno riferimento al primo e quelli relativi al secondo. Come per l'attributo precedente, maga è presente in gran parte delle tabelle e viene spesso usato per costituire la chiave primaria di queste. Ovviamente anch'esso non può assumere valori uguali a *null*.
ragg	varchar	si	Nel contesto in esame questo campo non ha un significato preciso. Fa parte, come i precedenti, della chiave primaria e gli deve quindi essere assegnato un valore diverso da *null* alla creazione di ogni record. Per ogni riga di Ordini_Testata, *ragg* contiene lo stesso valore del campo *numeOrdi*. L'unica differenza tra i due è il tipo dei dati contenuti. In questo caso si trattano varchar. *Ragg* è un chiaro esempio della ridondanza di dati precedentemente citata.
tipoOrdi	varchar	si	Definisce il tipo dell'ordine di un determinato record. Gli ordini possono essere di varia natura. Può essere generato un ordine di acquisto, di prelievo dal magazzino, di trasporto, di produzione ecc. L'unico tipo di ordine preso in considerazione per lo svolgimento dell'attività di stage è quello di produzione.

Campo	Tipo	Obbligatorio	Descrizione
annoOrdi	T_ANNO	si	Indica l'anno in cui è stato inserito l'ordine nel sistema. Ha uno scopo puramente indicativo dal momento che un altro campo (non descritto perché mai utilizzato durante il mio lavoro) indica la data precisa di creazione del record. Nonostante ciò, anche questo campo fa parte della chiave primaria della tabella e non può quindi essere *null*. Non è un dato di tipo numerico, data o stringa. In questo caso il tipo è stato definito da Madlab come T_ANNO il quale incapsula un dato numerico di 4 cifre.
numeOrdi	T_NUM_ORD	si	Il numero dell'ordine è il campo che, più di tutti gli altri precedentemente descritti, funge da chiave primaria, in quanto non esistono due record della tabella con *numeOrdi* di pari valore. A questo fa eccezione il campo *ragg* che però, come già accennato nella descrizione dello stesso, riporta sempre lo stesso valore di *numeOrdi* nel record rispettivo. Anche in questo caso il tipo è stato ridefinito ma, in definitiva, si tratta di un varchar di 11 caratteri.
viaggio	T_VIAGGIO	no	Come ripetuto più volte, Automa nasce come gestionale per il settore dei trasporti. Nella versione del programma personalizzata per Manitou, il significato di “viaggio” viene sovrascritto con quello di “lotto”, nonostante il nome di tale campo, nella maggioranza delle tabelle, sia rimasto inalterato. Ogni record di Ordini_Testata, fa capo ad uno ed un solo lotto. Il tipo di tale campo, pur essendo stato creato ad hoc, è un varchar di 10 caratteri.

A.2 Ordini_Dettaglio

Campo	Tipo	Obbligatorio	Descrizione
stab	varchar	si	Già descritto per la tabella Ordini_Testata. Il significato dei dati rimane lo stesso anche in questo caso.
maga	varchar	si	Già descritto per la tabella Ordini_Testata. Il significato dei dati rimane lo stesso anche in questo caso.
ragg	varchar	si	Già descritto per la tabella Ordini_Testata. Il significato dei dati rimane lo stesso anche in questo caso.
tipoOrdi	varchar	si	Già descritto per la tabella Ordini_Testata. Il significato dei dati rimane lo stesso anche in questo caso.
annoOrdi	T_ANNO	si	Già descritto per la tabella Ordini_Testata. Il significato dei dati rimane lo stesso anche in questo caso.
numeOrdi	T_NUM_ORD	si	Già descritto per la tabella Ordini_Testata. Il significato dei dati rimane lo stesso anche in questo caso.
rigaOrdi	varchar	si	Assieme a tutti i precedenti, questo campo definisce la chiave primaria di Ordini_Dettaglio ed è il solo che rende unico ogni record di tale tabella. Non può assumere valore *null*.
dataProd	datetime	no	La data di produzione indica il giorno per cui è previsto l'inserimento, nella rispettiva linea, del componente cui fa riferimento il record. Ciò che definisce la sequenza di entrata in linea di montaggio tra più elementi con *dataProd* uguale, è un altro campo (*sequProd*) non descritto nel presente documento in quanto mai utilizzato.

Campo	Tipo	Obbligatorio	Descrizione
ubicAttuale	varchar	no	Indica l'ubicazione attuale presso la quale risiede il componente descritto nel record. Nel caso in cui il componente non sia ancora entrato in linea di produzione, questo campo indica l'area del magazzino in cui è attualmente posizionato il pezzo. In alternativa, *ubicAttuale* definisce la specifica postazione in cui l'elemento verrà montato per diventare parte della commessa. Dal momento che il nome delle postazioni inizia con la lettera indicante la linea a cui queste appartengono, *ubicAttuale* è sufficiente a risalire a questo tipo di informazione senza dover coinvolgere altri campi o tabelle.

A.3 Viaggi

Campo	Tipo	Obbligatorio	Descrizione
stab	varchar	si	Già descritto per la tabella Ordini_Testata. Il significato dei dati rimane lo stesso anche in questo caso.
maga	varchar	si	Già descritto per la tabella Ordini_Testata. Il significato dei dati rimane lo stesso anche in questo caso.
anno	T_ANNO	si	Già descritto per la tabella Ordini_Testata. Il significato dei dati rimane lo stesso anche in questo caso.
viaggio	T_VIAGGIO	si	Come già spiegato in precedenza, il significato del campo è cambiato nonostante il nome sia rimasto inalterato. Viaggio rappresenta un lotto a cui possono essere associate una o più commesse. Il tipo di questo attributo non è predefinito. Si tratta infatti di un nuovo tipo di dato denominato T_VIAGGIO, il quale incapsula, al suo interno, un varchar di 10 caratteri. Assieme ai 3 campi precedenti, viaggio fa parte della chiave primaria e non può quindi essere *null*.

A.4 Giorni_Non_Lavorativi

Campo	Tipo	Obbligatorio	Descrizione
stab	varchar	si	Già descritto per la tabella Ordini_Testata. Il significato dei dati rimane lo stesso anche in questo caso.
maga	varchar	si	Già descritto per la tabella Ordini_Testata. Il significato dei dati rimane lo stesso anche in questo caso.
linea	varchar	si	Dal momento che la manutenzione di una linea, tiene bloccata solamente il segmento della catena di produzione su cui si effettua questo tipo di intervento, all'inserimento di un record di Giorni_Non_Lavorativi, bisogna indicare la linea per cui la data considerata non è lavorativa. Assieme a tutti gli altri campi della tabella, anche questo ne definisce la chiave primaria e non può quindi assumere valore *null*.
data	datetime	si	Indica semplicemente la data per cui è previsto che la linea indicata nello stesso record sia inattiva. I giorni in cui tutto lo stabilimento rimane chiuso (ad eccezione del sabato e della domenica che vengono già considerati giorni non lavorativi) saranno presenti in tanti record quante sono le linee di produzione. Dal momento che tutti gli altri campi della tabella possono comparire assieme più volte tra i record della tabella, anche questo attributo deve far parte della chiave primaria e deve pertanto assumere sempre valori diversi da *null*.

A.5 TMP_Gestione_Lotti

Campo	Tipo	Obbligatorio	Descrizione
stab	varchar	si	Già descritto per la tabella Ordini_Testata. Il significato dei dati rimane lo stesso anche in questo caso.
maga	varchar	si	Già descritto per la tabella Ordini_Testata. Il significato dei dati rimane lo stesso anche in questo caso.

Campo	Tipo	Obbligatorio	Descrizione
lotto	varchar	si	Indica il numero del lotto. Non essendoci la necessità di effettuare calcoli tra dati salvati in questo campo, il tipo scelto è letterale.
linea	varchar	si	Indica la linea presso la quale il lotto è stato o sarà accodato per la produzione.
host	varchar	si	Host indica il nome della workstation da cui si sta utilizzando l'applicazione. Come detto in precedenza, questo campo è utile al fine di tenere separati i dati nel caso di esecuzioni simultanee o alternate di più utenti da diverse postazioni.
dataIni	datetime	si	La data di inizio del lotto è la data con valore minore (cioè quella più vecchia) tra tutte le date di produzione delle componenti delle commesse relative al lotto in questione.
dataFine	datetime	si	La data di fine del lotto è la data con valore maggiore (cioè quella più avanti nel tempo) tra tutte le date di produzione delle componenti delle commesse relative al lotto in questione.
giorniEffettivi	int	si	Indica i giorni di cui il lotto necessita per essere terminato. Dal momento che tra la data di inizio (*dataIni*) e quella di fine (*dataFine*) del lotto possono esserci, oltre ai fine settimana, giorni nei quali la linea a cui viene assegnato il lotto in questione non è operativa, questo campo è necessario per tenere traccia del tempo reale impiegato per lo svolgimento del lavoro.
nuovoIni	datetime	no	Dopo che l'utente ha effettuato uno spostamento del lotto tramite la nuova interfaccia, la data di inizio sarà diversa. E' necessario salvare tale informazione in un nuovo campo per poter tornare più facilmente alla situazione di partenza. Nel caso di più spostamenti dello stesso lotto, *nuovoIni* viene di volta in volta sovrascritto con la nuova data di inizio scelta dall'utente. Se il lotto dovesse essere posizionato in una data non valida per la linea presso la quale è accodato, *nuovoInizio* viene sovrascritto con il primo giorno disponibile.

Campo	Tipo	Obbligatorio	Descrizione
nuovaFine	datetime	no	La descrizione del precedente attributo può essere adattata anche a questo tenendo solamente conto del fatto che, in questo caso, la nuova data considerata è quella di fine del lotto e non di inizio. Una volta calcolato il nuovo inizio evitando i giorni non validi, se il lotto dovesse trovarsi a cavallo di altre date presenti nella tabella Giorni_Non_Lavorativi per la stessa linea, *nuovaFine* dovrà essere incrementata di tanti giorni quanti quelli incontrati nel periodo occupato dal lotto.
modificato	bit	si	Modificato è semplicemente un flag che indica se il lotto considerato è stato spostato dall'ultimo caricamento dei dati dal database.
colore	varchar	no	Questo campo viene inizializzato durante l'assegnazione dei colori ai vari lotti, prima della visualizzazione degli stessi tramite l'interfaccia. Il valore del colore salvato è rappresentato su base esadecimale.

A.6 Log_Gestione_Lotti

Campo	Tipo	Obbligatorio	Descrizione
cont	int	si	Unica chiave primaria della tabella. *Cont* è semplicemente un contatore di tipo intero che viene incrementato automaticamente di un'unità ad ogni inserimento di un record nella tabella.
dataOra	datetime	si	Indica la data e l'ora, con precisione al millesimo di secondo, in cui è stata registrata l'operazione delicata.
host	varchar	si	Già descritto per la tabella TMP_Gestione_Lotti. Il significato dei dati rimane lo stesso anche in questo caso.
utente	varchar	si	Indica il nome del profilo dell'utente con cui è stato effettuato l'accesso al sistema operativo della workstation dalla quale è stata svolta l'operazione registrata.

Campo	Tipo	Obbligatorio	Descrizione
stab	varchar	no	Già descritto per la tabella Ordini_Testata. Il significato dei dati rimane lo stesso anche in questo caso.
maga	varchar	no	Già descritto per la tabella Ordini_Testata. Il significato dei dati rimane lo stesso anche in questo caso.
linea	varchar	no	Già descritto per la tabella TMP_Gestione_Lotti. Il significato dei dati rimane lo stesso anche in questo caso.
lotto	varchar	no	Già descritto per la tabella TMP_Gestione_Lotti. Il significato dei dati rimane lo stesso anche in questo caso.
dataIni	datetime	no	Già descritto per la tabella TMP_Gestione_Lotti. Il significato dei dati rimane lo stesso anche in questo caso.
dataFine	datetime	no	Già descritto per la tabella TMP_Gestione_Lotti. Il significato dei dati rimane lo stesso anche in questo caso.
nuovoIni	datetime	no	Già descritto per la tabella TMP_Gestione_Lotti. Il significato dei dati rimane lo stesso anche in questo caso.
nuovaFine	datetime	no	Già descritto per la tabella TMP_Gestione_Lotti. Il significato dei dati rimane lo stesso anche in questo caso.

A.7 Controlli principali

Istanza	Tipo	Descrizione
btnConn	commandButton	La pressione di questo tasto modifica lo stato attuale della connessione tra l'interfaccia ed il database. Nel caso in cui non sia attiva alcuna connessione, ne viene stabilita una. In caso contrario la connessione esistente cessa di esistere.

Istanza	Tipo	Descrizione
btnLoad	commandButton	Questo pulsante, premuto in un istante qualsiasi durante l'esecuzione dell'applicativo, effettua il caricamento dei dati che saranno poi visualizzati nel riquadro grafico. Alla pressione del tasto in questione, nel caso in cui esistano dati nella tabella temporanea, viene chiesto all'utente su quale insieme di dati desidera lavorare. Nel caso in cui il pulsante venga premuto quando sul riquadro grafico è già presente una situazione su cui lavorare, viene chiesta la conferma dell'operazione in corso da parte dell'utente in quanto, un ulteriore caricamento dei dati, sovrascriverebbe la situazione visualizzata.
btnSave	commandButton	La pressione di *btnSave* sovrascrive i dati presenti in Ordini_Dettaglio. In particolare, aggiorna i campi *dataProd* dei componenti facenti parte delle commesse che costituiscono i lotti movimentati tramite l'applicazione.
Form	form	*Form* è il contenitore di tutti gli altri controlli appartenenti all'applicazione. Si tratta semplicemente della finestra principale alla chiusura della quale, viene forzata l'eliminazione della connessione con il database e la deallocazione della memoria occupata dal programma in esecuzione.
Gantt	phGantX	Questo controllo è il fulcro dell'intero modulo. Una sua istanza fornisce le basi per gestire, in maniera strettamente correlata tra loro, alberi, liste, tabelle, oggetti temporali e vincoli di propedeuticità tra questi. Una porzione del riquadro visualizzato, una volta inserito il controllo nel *form*, rappresenta l'albero della struttura dello stabilimento. L'altra parte del riquadro contiene il diagramma il cui asse orizzontale rappresenta il tempo. Tale schema è diviso per righe, ognuna legata ad un nodo dell'albero citato. Su queste verranno disegnati gli elementi temporali rappresentanti i lotti di produzione.

A.8 Componenti visuali

Controllo	Tipo	Eventi ridefiniti	Funzionalità
cmdCentra	CommandButton	Click	Imposta la visualizzazione del grafico temporale posizionando il giorno attuale al centro dell'area disegnata
cmdConn	CommandButton	Click	Permette di effettuare la connessione/disconnessione tra il modulo ed il database
cmdGGNonValidi	CommandButton	Click	Apre un nuovo form dal quale è possibile gestire la tabella contenente i giorni non lavorativi suddivisi per linea (funzionalità non richiesta e non ancora utilizzabile)
cmdLoad	CommandButton	Click	Permette di caricare e visualizzare i dati contenuti nel database
cmdResetScaler	CommandButton	Click	Reimposta la visualizzazione del grafico temporale come se il modulo fosse stato appena avviato, posizionando quindi la data odierna all'estrema sinistra dell'area disegnata
cmdSave	CommandButton	Click	Aggiorna il database sovrascrivendo i dati in modo che rispecchino la situazione impostata dall'utente tramite il modulo di gestione del piano principale di produzione
cmdAllarga	CommandButton	Click	Raddoppia l'altezza della riga del grafico temporale selezionata, migliorandone la visualizzazione nel caso in cui vi siano molti lotti sovrapposti
cmdRestringi	CommandButton	Click	Dimezza l'altezza della riga del grafico temporale selezionata
cmdZoomIn	CommandButton	Click	Restringe l'intervallo di tempo visualizzato

Controllo	Tipo	Eventi ridefiniti	Funzionalità
cmdZoomOut	CommandButton	Click	Amplifica l'intervallo di tempo visualizzato
Form	Form	Load, MouseMove, Resize, Unload	E' la finestra principale, nella quale sono contenuti tutti gli altri controlli descritti
frmInfoLotto	Frame	-	E' un semplice riquadro nel quale sono raggruppate le etichette contenenti le informazioni relative al lotto selezionato
frmPrincipale	Frame	-	E' il riquadro contenente quasi tutti i controlli appartenenti a *Form*. Lo scopo di tale controllo è puramente estetico.
Gantt	phGantX	OnGantAreaDrawBackground, OnHintInfo, OnMouseMove, OnSelectionChangedGrid, OnValueChangedGantTime	E' il controllo, aggiunto tramite l'OCX *phGantTimePackage*, utilizzato per rappresentare graficamente la situazione delle linee di produzione facenti parte della struttura dello stabilimento del cliente. Sfruttando gli eventi messi a disposizione da questo controllo e ridefinendoli opportunamente, è possibile permettere la gestione del piano principale di produzione con il solo uso del mouse.
lblColore	Label	-	Campo testuale di sola lettura con funzionalità di legenda. Quando il puntatore del mouse passa sopra uno degli oggetti rappresentanti i lotti di produzione, lo sfondo di questa etichetta assume il colore del lotto considerato
lblFine	Label	-	Campo testuale di sola lettura rappresentante la data di terminazione del lotto su cui è attualmente posizionato il puntatore del mouse

Controllo	Tipo	Eventi ridefiniti	Funzionalità
lblGEffettivi	Label	-	Campo testuale di sola lettura rappresentante il numero di giorni occupati dal lotto su cui è attualmente posizionato il puntatore del mouse
lblGiorno	Label	-	Campo testuale di sola lettura rappresentante il giorno in corrispondenza del puntatore del mouse quando questo sia posizionato all'interno dell'area grafica
lblInizio	Label	-	Campo testuale di sola lettura rappresentante la data di inizio del lotto su cui è attualmente posizionato il puntatore del mouse
lblLinea	Label	-	Campo testuale di sola lettura rappresentante la linea assegnata al lotto su cui è attualmente posizionato il puntatore del mouse
lblMaga	Label	-	Campo testuale di sola lettura rappresentante il magazzino contenente la linea assegnata al lotto su cui è attualmente posizionato il puntatore del mouse
lblNumero	Label	-	Campo testuale di sola lettura rappresentante il numero di serie del lotto su cui è attualmente posizionato il puntatore del mouse
lblOra	Label	-	Campo testuale di sola lettura rappresentante l'ora in corrispondenza del puntatore del mouse quando questo sia posizionato all'interno dell'area grafica
lblStab	Label	-	Campo testuale di sola lettura rappresentante lo stabilimento nel quale dev'essere prodotto il lotto su cui è attualmente posizionato il puntatore del mouse

Controllo	Tipo	Eventi ridefiniti	Funzionalità
picConnHolder	PictureBox	-	Dal momento che la connessione e la disconnessione sono gestite da un unico pulsante la cui grafica cambia a seconda dello stato del modulo, questo controllo grafico permette di mantenere in memoria una delle due immagini senza dover accedere a file esterni
picDisconnHolder	PictureBox	-	Ha la stessa funzione del controllo precedente con la sola differenza che questo mantiene in memoria l'immagine alternativa
sbStatusBar	StatusBar	-	Barra di stato tramite la quale vengono visualizzate informazioni relative alle operazioni attualmente in esecuzione o già terminate
wcWheelCatcher	WheelCatcher	WheelRotation	Controllo nascosto tramite il quale è possibile catturare un evento scatenato dalla rotazione della rotellina del mouse. Nel modulo da me creato, al movimento di questa corrisponde lo slittamento dell'asse temporale

A.9 Funzioni in Visual Basic

Funzione	Tipo: tipo ritorno	Funzionalità
CalcolaAltezzaRiga(recordset)	Function: integer	Dal momento che, per ogni riga del grafico, non vi è un limite alle sovrapposizioni tra lotti, lavorare sull'altezza delle righe è indispensabile perché l'utente riesca a comprenderne il contenuto. Questa funzione, a seconda del numero ritornato dall'invocazione di *contaCollisioni*, ritorna il valore più adeguato da assegnare all'altezza della riga considerata

Funzione	Tipo: tipo ritorno	Funzionalità
connetti()	Function: boolean	Si occupa semplicemente di stabilire una connessione tra il modulo ed il database utilizzando specifici parametri salvati in un file esterno
contaCollisioni(long, recordset, long)	Function: integer	Invocata unicamente da *calcolaAltezzaRiga*, questa funzione rileva il massimo numero di sovrapposizioni tra lotti diversi, appartenenti alla medesima riga dell'area grafica. In questo modo è possibile capire quanto ampliare la visualizzazione di tale riga al fine di migliorarne la comprensione da parte dell'utente
disconnetti()	Subroutine: -	Distrugge la connessione esistente tra il modulo ed il database di Automa
getUserName()	Function: string	Ritorna il nome dell'utente connesso alla macchina da cui si sta utilizzando il modulo di gestione del piano principale di produzione
giornoValido(date, string, string, string)	Function: boolean	Ritorna false nel caso in cui la data, passata come parametro, sia un sabato, una domenica o un qualsiasi giorno non valido per la linea considerata (passata anch'essa come parametro)
initActivity(IphDataEntity_ Tree2, recordset)	Subroutine: -	Viene richiamata da *loadTreeItemsOwnedBy* per ogni linea dello stabilimento. Dopo aver inizializzato il nodo corrispondente alla linea corrente, richiama *loadTimeItemsOwnedBy* per riportare nel grafico gli oggetti rappresentanti i lotti associati a questa
initTimeItem(IphDataEntity_ GantTime2, recordset)	Subroutine: -	Viene richiamata da *loadTimeItemsOwnedBy* per ogni lotto esistente. Si occupa di associare univocamente le informazioni ricavate dal database, tramite la funzione che la invoca, relative al lotto corrente, all'oggetto grafico che lo rappresenta

Funzione	Tipo: tipo ritorno	Funzionalità
LoadTimeItemsOwnedBy(IphDataEntity_Tree2, variant)	Subroutine: -	Viene richiamata da *initActivity* alla fine del corpo di questa. Si occupa di caricare dal database le informazioni relative ai lotti associati alla linea attualmente in fase di inizializzazione. Per ognuno di tali lotti richiama a sua volta *initTimeItem* in modo da trasferire le informazioni riguardanti il lotto corrente, all'oggetto grafico che lo rappresenta
LoadTreeItemsOwnedBy(IphDataEntity_Tree2, variant)	Subroutine: -	Viene richiamata tante volte quanti sono i magazzini per lo stabilimento considerato. Inizializza il nodo corrispondente al magazzino corrente con le informazioni utili ed infine richiama *initActivity* per ogni linea appartenente a questo.

Glossario

B

Bordo-linea: (nel contesto in esame) zona di limitato stoccaggio temporaneo utile a velocizzare il processo di produzione eliminando i tempi morti dovuti all'attesa del prelievo dei pezzi necessari direttamente dal magazzino.

Buffer: in informatica è una zona di memoria implementabile sia tramite hardware che tramite software usata temporaneamente per l'input o l'output dei dati oppure per velocizzare l'esecuzione di alcune operazioni di comunicazione fra componenti che lavorano a velocità
differenti.

Business plan: riassunto di come un imprenditore o un manager intende organizzare un'attività imprenditoriale e implementare attività necessarie e sufficienti alla sua buona riuscita.

C

Chiave esterna: campo di una tabella appartenente ad un DBMS avente la funzione di relazionare la tabella di cui fa parte con le altre tabelle che costituiscono il database.

Chiave primaria: insieme di campi che permette di identificare univocamente un record in una tabella di un database.

Commessa: (nel contesto in esame) macchina la cui produzione è ordinata dall'acquisto da parte di un cliente.

Codice a barre: insieme di elementi grafici a contrasto elevato disposti in modo da poter essere facilmente letti da un sensore e decodificati tramite un apposito circuito integrato.

Crash: blocco o l'improvvisa chiusura, non richiesta, di un programma, oppure il blocco completo dell'intero sistema operativo.

D

DBMS (DataBase Management System): sistema software progettato per consentire la creazione e manipolazione efficiente di database.

Diagramma di Gantt: strumento di supporto alla gestione dei progetti, così chiamato in ricordo dell'ingegnere statunitense che si occupava di scienze sociali che lo ideò nel 1917,Henry Laurence Gantt(1861 – 1919).

Dispositivo di identificazione automatica: sistema hardware per il riconoscimento automatico di oggetti o persone in base a caratteristiche innate o artificialmente applicate.

Distinta base: (in inglese BOM – Bill of Material) descrive in prodotto in termini di sistemi, sottosistemi e componenti elementari. E' rappresentabile come la lista delle parti utilizzate per progettare e costruire un prodotto.

E

E/R: il modello entità-relazioni (E/R) viene adoperato per analizzare le caratteristiche di una situazione, prescindendo dagli eventi che si verificheranno. Ciò al fine di costituire un modello concettuale di dati sentito come indipendente dalle applicazioni. Il modello entità associazione si distingue per la presenza di vari elementi

che lo definiscono:
1. Le entità, oggetti che avranno un particolare senso e vengono classificati secondo criteri particolari;
2. Le associazioni, legami che si formano tra due entità;
3. Gli attributi, caratteristiche che qualificano e contrassegnano le entità.

Event driven: stile di programmazione in cui i vari blocchi di codice vengono eseguiti in risposta a determinati *eventi*, od azioni, su controlli dotati di rappresentazione grafica.

F

Flag: variabile che può assumere solo due stati ("vero" o "falso", "on" e "off", "1" e "0", "acceso" e "spento") e che segnala, con il suo valore, se un dato evento si è verificato oppure no, o se il sistema è in un certo stato oppure no.

Form: nell'ambiente di sviluppo Visual Basic indica la finestra principale dell'applicazione, in cui possono essere inseriti elementi visuali come pulsanti o caselle di testo.

I

Isola: (nel contesto in esame) linee costituite da una singola postazione in cui vengono costruiti componenti complessi utili all'assemblaggio finale del prodotto che ne richiede la presenza.

K

Kit: (nel contesto in esame) elemento semi-assemblato comune a varie macchine producibili.

L

Linea di produzione (o di montaggio): processo di assemblaggio utilizzato nelle moderne industrie teso ad ottimizzare il lavoro degli operai e a ridurre i tempi necessari per il montaggio di un manufatto complesso.

Log: registro cronologico di eventi sotto forma di file sequenziale sempre aperto in scrittura, che viene chiuso e conservato a cadenze regolari o di segmento di base dati con accesso diretto mediante chiave cronologica.

Lotto: (nel contesto in esame) insieme di commesse.

M

Magazzino automatico/verticale: impianto automatizzato preposto allo stoccaggio delle merci.

Mancante: (nel contesto in esame) prodotto finito non completo a causa del mancato montaggio di uno o più componenti nella fase di assemblaggio.

Missione: (nel contesto in esame) struttura dati di vario tipo utile a gestire la “prenotazione” di articoli richiesti per la produzione di una commessa o un lotto al fine di evitare carenze impreviste, velocizzare e comandare l'attività dei magazzinieri, guidare l'assemblaggio compiuto dalle diverse postazioni, minimizzare gli errori o semplificarne la gestione e velocizzarne la risoluzione.

MPS (Master Production Schedule): vedere Piano Principale di produzione.

O

OCX (OLE Control eXtension): componente utilizzato in Visual Basic ed altri linguaggi di programmazione per aggiungere potenzialità a tali strumenti.

OLE (Object Linking and Embedding): tecnologia per la creazione di documenti composti sviluppata da Microsoft.

OptionButton: (pulsante di opzione) consente di visualizzare un'opzione che può essere attivata o disattivata.

Ordine di dettaglio: (nel contesto in esame) record che lega una particolare commessa ad uno dei componenti (con relativa quantità da prelevare) necessario alla costruzione di questa.

Ordine di testata: (nel contesto in esame) record del database che associa il cliente che ha richiesto la produzione di una commessa alla commessa vera e propria.

P

Pallet: (pedana o bancale) attrezzatura utilizzata per l'appoggio di vari tipi di materiale, destinati ad essere immagazzinati e movimentati tramite attrzzature specifiche e trasportati con vari mezzi di trasporto.

Panel: rappresenta un singolo pannello dell'insieme Panels di un controllo StatusBar. Può contenere testo e immagini che visualizzano lo stato di un'applicazione.

Panels: contenitore di oggetti di tipo Panel.

Piano principale di produzione: programma di fabbricazione che quantifica processi significativi, compiti ed altre risorse allo scopo di ottimizzare la produzione, identificare i colli di bottiglia, ed anticipare bisogni e beni finiti.

Pixel: elemento puntiformi che compone la rappresentazione di un'immagine nella memoria di un computer.

Pop-up: form funzionalmente marginale o a scopo informativo richiamato e visualizzato da un programma in esecuzione.

Postazione: (nel contesto in esame) porzione della linea di produzione in cui viene eseguita una specifica fase di costruzione della macchina.

Premontaggio: (nel contesto in esame) componente complesso assemblato (presso un'isola ben definita) in stock o in parallelo alla commessa che ne richiede la presenza.

Provider: azienda/organizzazione che fornisce un servizio.

R

Record: oggetto di un database strutturato in dati che contiene un insieme di campi o elementi, ciascuno dei quali possiede nome e tipo propri.

Recordset: oggetto di Visual Basic rappresentante i record di una tabella base o i record che risultano dall'esecuzione di una query.

Riga d'ordine: (nel contesto in esame) record del database che associa l'ordine di una commessa da parte di un cliente ad uno dei kit che la compongono.

S

Scarico: (nel contesto in esame) cancellazione dal sistema di articoli non più utilizzabili all'interno del magazzino perché spediti o montati in una macchina in costruzione.

Semi-assemblato: prodotto in costruzione.

Simulazione: (nel contesto in esame) previsione sull'esito della produzione di un lotto in termini di mancanti a fronte delle momentanee giacenze presenti nel magazzino (considerato come ogni area utilizzabile nello stabilimento per lo stoccaggio di merci).

Stored procedure: programma scritto in SQL od in altri linguaggi e mantenuto nel database stesso.

Swap: letteralmente “scambio”.

T

Thread: suddivisione di un programma in due o più operazioni che vengono eseguite in modo concorrente.

Trasloelevatore: particolare robot a tre assi utilizzato in ogni magazzino automatico. È composto da un elevatore traslante che si muove su rotaia, al cui interno vi è l'organo di presa (solitamente forcole telescopiche).

U

Udc (Unità di carico): la più piccola unità di trattamento degli articoli gestita da Automa. Può contenere diversi pezzi, prelevabili in quantità variabile di cui non viene tenuto conto fino all'esaurimento della giacenza nell'udc stessa.

UML (Unified Modeling Language): linguaggio di manipolazione orientato agli oggetti studiato per descrivere l'architettura di un sistema in dettaglio.

Use-case: diagrammi dedicati alla descrizione delle *funzioni* o *servizi* offerti da un sistema, così come sono percepiti e utilizzati dagli attori che interagiscono col sistema stesso.

W

Wireless: termine usato per indicare sistemi di comunicazione tra dispositivi elettronici, che non fanno uso di cavi.

Workstation: generalmente, computer fisso ad alte prestazioni, utilizzato da professionisti per il lavoro su specifici servizi o compiti.

Bibliografia

Testi consultati:

Logistica:

- Fabrizio Dallari, "MPH Manuale di logistica", Università C. Cattaneo di Castellanza (2004)

SQL:

- Giuseppe callegarin, "Nuovo corso di informatica – Basi di dati e sistemi informativi" Vol.3, CEDAM (1996)

Siti Internet consultati:

Visual Basic:

- http://msdn.microsoft.com/it-it/library/2x7h1hfk.aspx

Transact-SQL:

- http://msdn.microsoft.com/it-it/library/ms189826(SQL.90).aspx

OCX phGantTimePackage:

- http://www.plexityhide.com

Il cliente Manitou:

- http://www.it.manitou.com

L'azienda ospitante lo stage:

- http://www.madlab.biz

Altre fonti:

- http://www.wikipedia.it
- http://www.wikipedia.org