Università degli Studi di Padova

Facoltà di Scienze MM.FF.NN.
Laurea in Informatica

Progettazione e sviluppo di un ambiente CAD

Stage presso IT+Robotics

Candidato: **Mirko Polato** [579350]

Relatore: Ch.mo Prof. **Paolo Baldan**

Tutor Esterno: Alberto Conz

Anno accademico 2010/2011

Alla mia famiglia e
ad Alessandra.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Introduzione

Il progresso tecnologico in ambito industriale ha raggiunto oggi un notevole sviluppo. Molti dei processi meccanici e ripetitivi sono stati automatizzati da robot manipolatori, portando un conseguente aumento della richiesta di software dedicato. La predisposizione e la programmazione di questi impianti è un processo tanto oneroso quanto delicato, è necessario quindi che il software offra un ambiente di simulazione, il quale produca in output il codice per la programmazione dei microcontrollori installati nei robot. A tal scopo, dal 2008 l'azienda IT+Robotics (Spin Off dell'Università degli Studi di Padova) sta sviluppando un progetto denominato SimulEasy v2, che consiste in una serie di soluzioni software, utili per la simulazione virtuale di celle di lavoro automatizzate in cui siano presenti robot manipolatori. L'applicazione principale di questo progetto è WorkCellSimulator, composta a sua volta da più componenti specifiche, che vanno dalla creazione della cella di lavoro simulata, alla creazione del codice macchina che esegue nella realtà ciò che è stato ottenuto nella simulazione.

Questo stage ha lo scopo di ampliare uno dei componenti ancora in fase embrionale, nello specifico l'editor *CAD*[1]. L'ambiente CAD (Cad Editor) sviluppato deve permettere il disegno bidimensionale di parti delle celle di lavoro e dei componenti manipolati al suo interno. Questi modelli 2D verranno successivamente estrusi per ottenere i corrispondenti modelli tridimensionali che saranno utilizzati nel simulatore.

L'applicazione deve poter offrire un set limitato di funzionalità, che permettano il disegno di modelli complessi in modo semplice e intuitivo. Cad Editor non vuole in alcun modo competere con ambienti professionali quali AutoCAD o ArchiCAD, ma vuole essere un CAD alternativo e allo stesso tempo compatibile con tali ambienti. La compatibilità è data dall'utilizzo di un formato particolare (*DXF*) per i file, i quali possono essere aperti e modificati da tutte le applicazioni CAD prodotte

[1] I termini scritti in *corsivo* sono chiariti nel Glossario alla fine di questa tesi.

da AutoDesk ®.

Durante lo sviluppo di un ambiente CAD, le problematiche che si devono affrontare sono di natura molto variabile. Prima tra tutte, la gestione dell'area di disegno, che deve essere precisa e veloce nel *rendering* 2D, garantendo una certa fluidità durante la creazione del modello. La natura geometrica e algebrica di quasi la totalità delle funzioni rappresenta un'altro problema non banale. Le soluzioni devono essere le più semplici possibili, in modo che la computazione non sia troppo pesante causando ritardi durante il ridisegno dell'area di lavoro.
Essendo poi Cad Editor un software già parzialmente implementato, è indispensabile capire quali parti possono essere riutilizzate e quali strutturalmente e/o formalmente devono essere riviste.

L'analisi, in tal senso, ha rappresentato una fase fondamentale, portando alla luce le lacune di Cad Editor. Molte delle funzionalità di creazione sono state aggiunte o reimplementate, come ad esempio la possibilità di creare un arco o un cerchio dati 2 punti. Anche le funzioni di modifica sono state notevolmente integrate, con l'aggiunta delle funzioni "offset", "raccorda" e "specchia". La funzione "offset", tra tutte, è la funzione più utilizzata, poiché permette con pochi click di creare linee parallele e cerchi concentrici. La funzione "specchia" diventa molto utile quando il modello da disegnare è simmetrico. In questi casi infatti, è sufficiente creare solo una metà del modello, la quale viene specchiata, e la parte mancante viene così completata.
A differenza delle precedenti, la funzione "raccorda" è molto meno duttile, poiché esegue esclusivamente l'arrotondamento dell'angolo formato da 2 linee che si intersecano. Nonostante sia molto mirata, agevola un operazione che altrimenti sarebbe molto complessa da effettuare manualmente.
Oltre alle mancanze funzionali, dall'analisi sono emerse molte carenze architetturali di Cad Editor. Questo ha portato alla scelta di eseguire un refactoring preliminare, con l'intento di delineare una struttura adeguata e semplice da ampliare. L'operazione di refactoring ha avuto un forte impatto nello sviluppo del CAD, cambiando considerevolmente la logica alla sua base.
Per quanto riguarda l'usabilità del software, è stata realizzata una maschera per l'inserimento dei dati tramite tastiera, caratteristica indispensabile per il disegno di

precisione. Infatti, il disegno di modelli si basa su dimensioni specifiche, l'utilizzo del solo mouse renderebbe questa operazione troppo difficile e dispendiosa. Come ultima attività, ma non per questo la meno importante, è stato effettuato il debug e l'integrazione della gestione dei livelli, fondamentali durante l'*estrusione*, poiché a livelli diversi corrispondono quote diverse. La possibilità di poterli colorare con tonalità differenti rende il modello molto più leggibile.

Il linguaggio di programmazione utilizzato per la realizzazione di questo progetto è il C#. Per agevolare la gestione dell'area di lavoro Cad, l'azienda ha reso disponibile la libreria di funzionalità CadLib, che riveste un ruolo centrale nello sviluppo dell'applicazione. L'intero progetto è stato realizzato sotto Windows, utilizzando Visual Studio 2008 come ambiente di sviluppo.

1.1 Organizzazione del documento

Il resto della tesi contiene: nel capitolo 2 lo studio del problema in cui vengono discusse le tecnologie utilizzate e la struttura della parte di Cad Editor implementata prima dello stage. Nel capitolo 3 è presentata l'analisi dei requisiti dell'applicazione, la quale rappresenta la base per la progettazione che è invece descritta nel capitolo 4. Nel capitolo 5 sono illustrate le fasi di test eseguite, infine l'ultimo capitolo contiene le conclusioni sullo stage.

CAPITOLO 2

Studio del Problema

In questo capitolo verrà descritta la libreria CadLib, utilizzata per lo sviluppo dell'applicazione, valutandone pregi e difetti. Verranno inoltre analizzate la struttura della parte di Cad Editor già implementata e il linguaggio di programmazione C#. Infine, saranno discusse le soluzioni CAD esistenti nel mercato, in modo da capire quali funzionalità devono essere perfezionate o sono del tutto mancanti in Cad Editor.

2.1 Il linguaggio C#

Il linguaggio usato per realizzare Cad Editor è C#. Il C# è un linguaggio di programmazione orientato agli oggetti puro, sviluppato da Microsoft® all'interno dell'iniziativa *.NET*. Nel 2001 è divenuto standard ECMA e successivamente, nel 2003, anche standard *ISO* (ISO/IEC 23270). L'utilizzo di questo linguaggio per realizzare l'applicazione è stato espressamente richiesto dal committente. Nonostante questo, l'azienda vede in C# un' ottima alternativa, ad alto livello, a Java. Inoltre, grazie all'ambiente di sviluppo integrato Visual Studio, sia la scrittura di codice che il debug vengono semplificati e ottimizzati.
Esistono però alcuni limiti che questo tipo di tecnologia, con Framework .NET 2.0, pone:

- Non è permesso effettuare ereditarietà multipla;
- Non esiste una struttura dati simile alla lista circolare;

2.2 La libreria CadLib

Per rendere meno costosa, in termini di tempo, e impegnativa la creazione dell'ambiente CAD, l'azienda ha acquistato la libreria CadLib.
CadLib è una libreria di funzionalità CAD proprietaria, rilasciata da Wout Ware

®. Oltre ad aggiungere moduli CAD alle applicazioni .NET, CadLib facilita notevolmente la manipolazione di file DXF (formato per modelli CAD di AutoDesk) e offre inoltre una sotto libreria di funzioni matematiche complesse. La scelta da parte dell'azienda di usufruire di questo strumento è dovuta da vari fattori:

- costo non troppo elevato;
- ampia documentazione con esempi di utilizzo;
- supporto tecnico molto valido;
- libreria in continua evoluzione.

La versione della libreria utilizzata all'inizio dello stage era la 1.0, durante lo svolgimento è stato eseguito l'upgrade alla versione 2.0.

2.2.1 Struttura di CadLib

Questa libreria è suddivisa in diversi namespace, ognuno avente le proprie caratteristiche peculiari. I namespace più utili per la realizzazione di Cad Editor sono elencati di seguito:

- **WW.Cad.Model**: in questo namespace la classe principale è DxfModel, che rappresenta il modello contenente le entità che si trovano nel disegno Cad. Infatti, all'interno di esso c'è un sotto namespace, **WW.Cad.Model.Entities**, che include tutte le entità supportate dalla libreria.
- **WW.Cad.Drawing**: questo namespace permette il disegno di un DxfModel sulla grafica. Attualmente solo il disegno *GDI* è supportato.
- **WW.Cad.IO**: contiene tutte le classi utili alla lettura e scrittura di file .dxf e .dwg. Queste classi permettono anche l'esportazione del modello in diversi formati immagine, tra cui png, gif, jpg e svg. É supportata anche l'esportazione in formato pdf.
- **WW.Math**: namespace contenente le strutture dati utili per il calcolo algebrico e geometrico.

2.2.2 Modelli DXF

Un modello DXF, ovvero un oggetto della classe DxfModel, rappresenta una concretizzazione del formato di dati ideato da AutoDesk, per permettere la manipolazioni di modelli CAD al di fuori di AutoCAD. Nella più semplice delle interpretazioni, un DxfModel è un contenitore di entità. Queste entità possono appartenere ad uno specifico livello (layer), oppure possono far parte del livello di default denominato layer 0. All'interno di un modello DXF sono racchiuse molte altre informazioni riguardanti gli stili, il sistema di coordinate e molto altro, che per ora non sono utili all'analisi.

2.2.3 Entità

Un'entità rappresenta un oggetto elementare o derivato con particolari caratteristiche di visualizzazione. Esempi di entità elementari lo sono le linee, i punti, i cerchi. Mentre per entità derivate (o composte) si intendono ad esempio le polilinee, i rettangoli e le quote. Quest'ultime possono essere rappresentate da più entità elementari che seguono particolari regole. Le entità rappresentano l'oggetto più piccolo contenuto in un modello. CadLib offre un set di entità precostruite che si trovano all'interno del namespace WW.Cad.Model.Entities.

2.2.4 Graphics GDI

Il solo modello DXF non è sufficiente per essere visualizzato, c'è bisogno di un gestore grafico rappresentato da un istanza dell'oggetto GDIGraphics3D, contenuto nel namespace WW.Cad.Drawing.GDI. Questo oggetto, grazie alle chiamate alle API di Windows, renderizza il modello CAD in 2D. Inoltre, offre una serie di funzioni utili per il disegno interattivo. Con interattivo si intende quando l'utente sta creando o trasformando un'entità. Quest'ultima, durante la sua manipolazione è in continua evoluzione, grazie a queste funzioni di disegno dinamico viene aumentata l'efficienza e la fluidità dell'applicazione, poiché evitano il ridisegno di tutto il modello.

2.2.5 Interazioni

In un ambiente CAD la gestione delle interazione dell'utente con la grafica è fondamentale. A questo scopo, CadLib offre l'interfaccia IInteractor, che deve essere

implementata da ogni classe che rappresenta un interattore utente-grafica. I metodi dell'interfaccia che devono essere implementati sono:

- ProcessMouseDown: rappresenta l'evento di pressione di un tasto del mouse;
- ProcessMouseUp: rappresenta l'evento di rilascio di un tasto del mouse;
- ProcessMouseWhell: rappresenta l'evento di rotazione della rotellina del mouse;
- ProcessMouseMove: rappresenta l'evento di spostamento del mouse;
- Activate: metodo richiamato all'attivazione dell'interattore. Solitamente viene utilizzato per effettuare inizializzazioni.
- Deactivate: metodo richiamato alla disattivazione dell'interattore.

2.2.6 Sommario

CadLib rappresenta un ottima libreria di supporto, ricca di funzionalità che facilitano notevolmente lo sviluppo di un ambiente CAD. Molto semplice da utilizzare ed estendere, con una documentazione davvero molto ampia e chiara anche se solo in lingua inglese. Il supporto online è efficiente e spesso efficace. Una nota negativa lo sono le entità, che vengono caratterizzate da proprietà a volte difficilmente manipolabili o che rendono ardua la modifica personalizzata delle stesse. Essendo CadLib in continua evoluzione, non sono rari i riscontri di piccoli bug all'interno delle funzionalità offerte.

2.3 Analisi di Cad Editor

In questa sezione viene analizzata la struttura esistente di Cad Editor, facendo le dovute considerazioni su come è stata progettata l'applicazione. Viene innanzitutto presentata l'interfaccia grafica e successivamente i vari componenti interni.

2.3.1 Interfaccia Grafica

La GUI di Cad Editor è composta da una *ribbon* contenente tutti i pulsanti utili per attivare le varie funzionalità dell'applicazione. La ribbon è composta da un unica scheda, questo non per scelta, ma per poter integrare Cad Editor come plugin di WorkCellSimulator. I pulsanti sono raggruppati in base alle loro caratteristiche. Ci sono principalmente 5 gruppi di pulsanti:

1. Disegno: sono i pulsanti che attivano la creazione di un'entità;
2. Misura: sono i pulsanti che permettono di effettuare misurazioni;
3. Stampa: sono i pulsanti che avviano la stampa o l'esportazione del modello 2D nei formati indicati precedentemente;
4. File: sono i pulsanti per scrivere/leggere da file. Nella versione plugin è presente il tasto di estruzione del modello;
5. Varie: questi pulsanti non hanno una categorizzazione specifica. La maggior parte rappresentano funzionalità di modifica o manipolazione delle entità.

Al di sotto della ribbon, al centro, si trova il *View Control*, ovvero l'area di lavoro CAD. Di default quest'area ha uno sfondo nero e le linee di disegno sono di colore bianco. Nel lato sinistro del View Control è posta la griglia delle proprietà, che contiene le caratteristiche notevoli delle entità selezionate. Queste proprietà possono essere modificate agendo direttamente sulla griglia. A destra infine, troviamo il pannello dedicato alla gestione dei livelli (Figura 2.1).

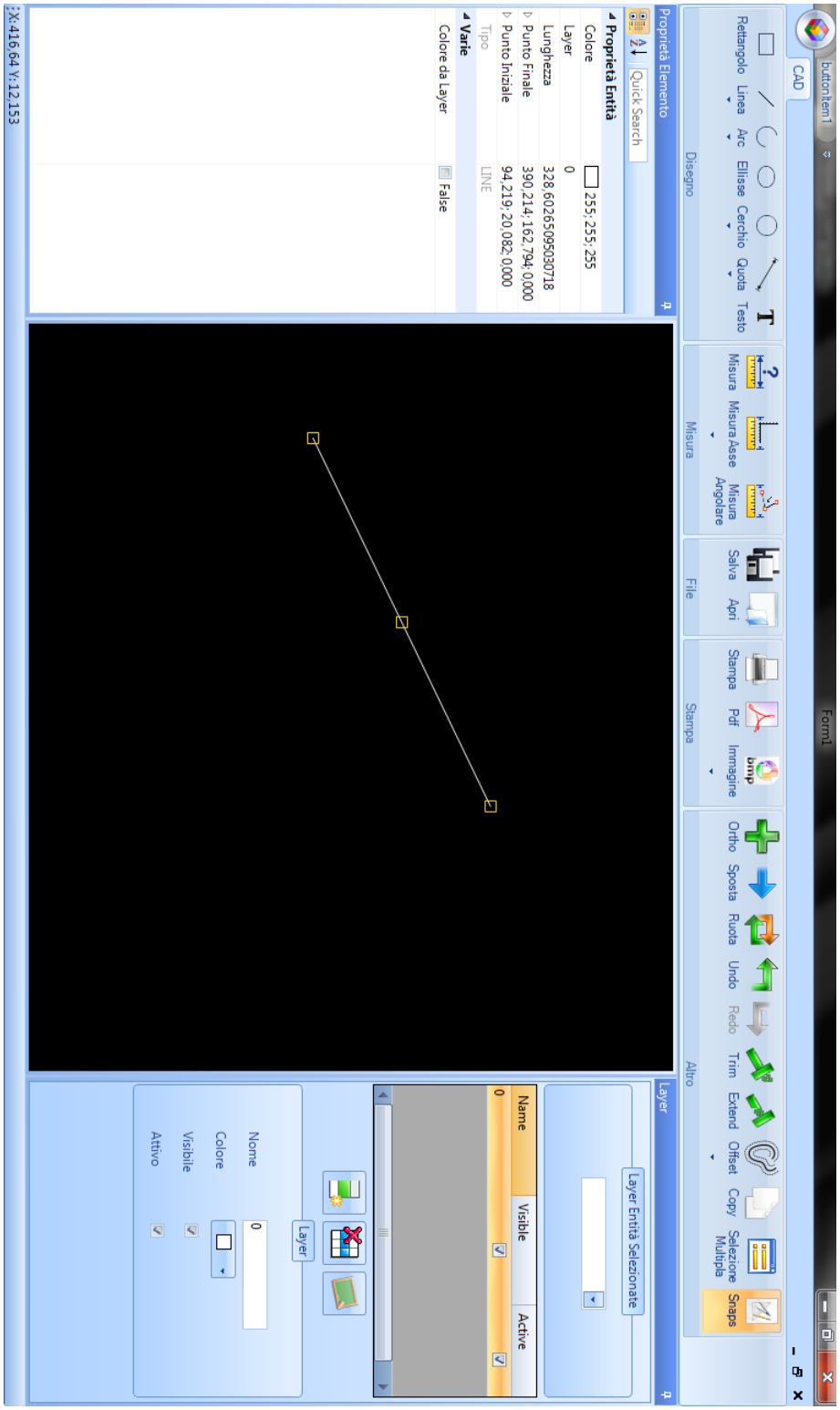

Figura 2.1: Interfaccia grafica di Cad Editor.

2.3.2 Classe View Control

Una delle classi fondamentali che compone Cad Editor è View Control.
View Control rappresenta l'area di lavoro su cui l'utente creerà i propri modelli CAD. A livello grafico è contraddistinta da un area a sfondo nero, su cui il puntatore prende una forma a croce. La sua funzione principale è quella di gestire gli eventi del mouse che avvengono su di essa, delegando alle classi specializzate il lavoro a più basso livello.
In figura 2.2 è illustrato il diagramma di collaborazione che rappresenta, in parte, cosa avviene durante la creazione di una linea.

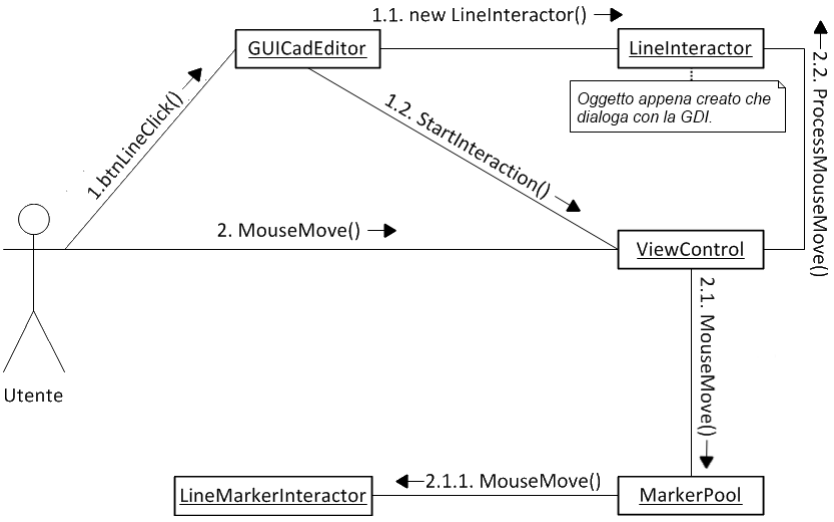

Figura 2.2: Diagramma di collaborazione: creazione linea.

Il diagramma si limita a raffigurare ciò che accade una volta premuto il tasto di creazione linea e successivamente spostato il mouse all'interno del View Control. La prima azione, è la pressione del tasto di creazione linea da parte dell'utente. Questo provoca la creazione di un nuovo interattore (Sez. 2.3.3), e viene avvisato il View Control che le operazioni che avverranno su di esso a partire da ora riguardano un interazione (StartInteraction), nel caso specifico la creazione di una linea. Ora l'utente, spostando il mouse sul View Control, causa una cascata di eventi che parte dal MouseMove del View Control stesso. Quest'ultimo, delega all'interattore la gestione dell'operazione dell'utente e avvisa il MarkerPool (Sez. 2.3.4) che è avvenuto uno spostamento del mouse. Quest'ultimo avviso è utile per l'eventua-

le attivazione di alcuni marker, che verranno trattati più in dettaglio nella sezione 2.3.4.

2.3.3 Interattori

Come detto precedentemente (Sez. 2.2.5), per gestire le interazioni dell'utente con l'ambiente grafico, bisogna implementare l'interfaccia IInteractor offerta da CadLib. Sostanzialmente, gli interattori, si occupano di gestire una data sequenza di operazioni, con lo scopo di creare (o modificare) un insieme di entità. Riprendendo l'esempio di Figura 2.2, si ha che l'interattore è di tipo LineInteractor, ovvero si occupa del disegno di una nuova linea. LineInteractor si aspetta che l'utente selezioni i 2 punti limite della linea, facendo 2 click all'interno del View Control nelle posizioni desiderate. Oltre a gestire la pressione del tasto sinistro del mouse, l'interattore gestisce anche lo spostamento del mouse, in modo da visualizzare l'anteprima della linea in fase di creazione. Ogni tipo di interattore si aspetta una sequenza di operazioni diverse, ma concettualmente il meccanismo di funzionamento è lo stesso per tutti.

Esiste una categoria di interattori particolare, i MarkerInteractor. Questi sono attivati tramite l'interazione con i marker, che verranno spiegati nella sezione 2.3.4, e si occupano dell'attivazione dell'evento di modifica delle entità. Una differenza fondamentale dai precedenti interattori, è che non implementano l'interfaccia IInteractor. Questo è dovuto al fatto che gestiscono unicamente il click sul marker, il resto dell'interazione e rimandata agli interattori descritti precedentemente.

2.3.4 Classe MarkerPool

Innanzitutto vediamo cosa si intende per *punti notevoli* di un'entità, detti anche *marker*. Ogni entità ha alcune caratteristiche particolari che devono poter essere modificate in modo semplice e intuitivo. Una linea, ad esempio, deve poter essere allungata o spostata con pochi click. Questa possibilità è data dai marker, ovvero dei piccoli quadrati posti nei punti di interesse dell'entità. Nella linea questi punti notevoli sono posti nel punto iniziale, finale e nel punto medio (Figura 2.3).

L'interazione con questi marker, attraverso i MarkerInteractor, provoca l'attivazione immediata, della modifica della linea. I marker hanno anche una funzione di

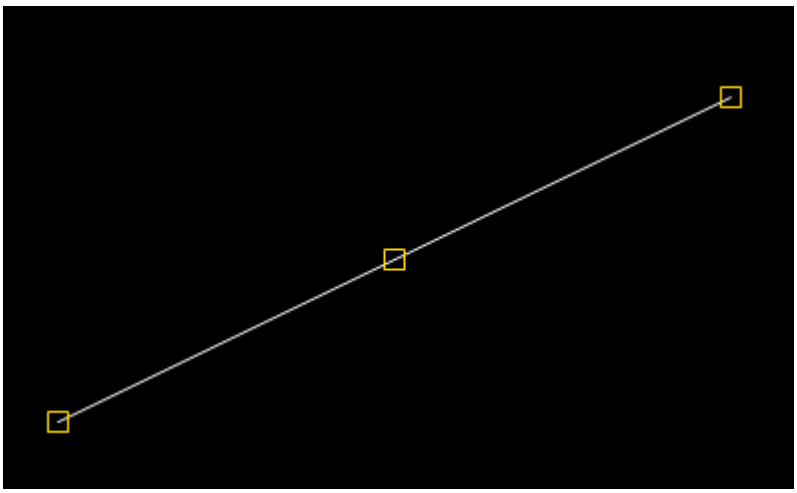

Figura 2.3: Punti notevoli di una linea.

aggancio per la creazione di altre entità.

La classe MarkerPool rappresenta un contenitore di MarkerInteractor, uno per ogni tipo di entità. Una volta che View Control avvisa il MarkerPool di un nuovo evento del mouse, quest'ultimo, in base all'evento, si comporta di conseguenza, attivando un MarkerInteractor o semplicemente visualizzando i marker vicini alla posizione corrente del mouse. Quindi, non sono le entità a definire i loro punti notevoli, ma i loro MarkerInteractor.

2.3.5 Classe DrawPanel

L'inserimento, da parte dell'utente, di una misura o di uno specifico punto, può essere problematico se eseguito attraverso l'uso del solo mouse. Per questo motivo Cad Editor permette l'inserimento attraverso una maschera, avente delle textbox, che segue il puntatore del mouse quando posto nel View Control. Questa maschera è concretizzata dalla classe DrawPanel, che graficamente appare come in Figura 2.4. Nonostante faciliti l'inserimento da tastiera, questo strumento è insoddisfacente, sia a causa dell'implementazione non del tutto corretta sia per l'effetto grafico spesso invadente e poco fluido. Per questi motivi, l'azienda ha espressamente richiesto la sostituzione di questo componente con una maschera a comparsa denominata EntityControl, la quale verrà discussa successivamente nel capitolo di Progettazione.

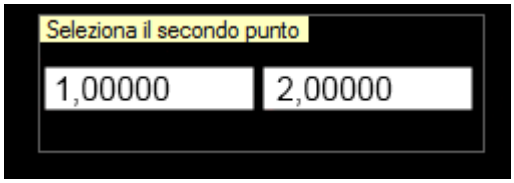

Figura 2.4: Aspetto del DrawPanel.

2.3.6 Struttura generale delle classi

Dal punto di vista architetturale, Cad Editor, è composto da 2 classi e 2 gerarchie fondamentali: CadEditorGUI, View Control e le gerarchie degli interattori (standard e marker). Sebbene le strutture gerarchiche siano ragionevoli, le soluzioni implementate per gestire le funzionalità, generalmente non rispettano il paradigma della programmazione ad oggetti. Spesso accade che una funzione accetti un'entità generica per poi filtrarla, tramite costrutti quali switch o if-else if, e specializzarsi nell'entità particolare. Questa soluzione è inaccettabile, poiché dovrebbe essere l'entità stessa a preoccuparsi di una sua eventuale modifica o manipolazione. A tal proposito, è da notare che non esiste alcuna classe che rappresenti una data entità. Vengono utilizzate direttamente le classi offerte dalla libreria CadLib, le quali offrono poca flessibilità e, in alcuni casi, non sono di semplice utilizzo.
Queste osservazioni hanno portato alla scelta di effettuare come primo processo, un refactoring dell'applicazione, che porterà alla struttura descritta nel capitolo di Progettazione.

2.3.7 Funzionalità di Cad Editor

La tabella 2.1, schematizza le funzionalità implementate in Cad Editor, descrivendole brevemente e specificandone il livello di completezza. Il 100% di completamento non indica che questa funzionalità non sia stata rivista. Per ulteriori informazioni di veda la tabella 4.1.

Funzione	Descrizione	Completa (%)
Creazione		
Arco	Arco dato il centro e 2 angoli	100
Cerchio	Cerchio dato centro e raggio	100
Rettangolo	Rettangolo dati 2 angoli opposti	100
Ellisse	Ellisse dati centro, asse maggiore e minore	100
Linea	Linea dati 2 punti limite	100
Polilinea	Multilinea dati più punti	100
Testo	Testo dati origine, rotazione e contenuto	100
Manipolazione		
Trim	Taglio di una linea	70
Extend	Estensione di una linea	100
Move	Spostamento di un'entità	100
Copy	Copia di un'entità	100
Rotate	Rotazione di un'entità	50
IO		
Export Img	Esportazione in formato immagine	90
Export PDF	Esportazione in formato PDF	100
Export DXF	Esportazione in formato DXF	100
Stampa	Stampa del modello	90
Open DXF	Apertura file DXF	60
Extrude	Estrudione modello in 3D	100
Altre		
Misura lineare	Misura lineare	100
Misura asse	Misura lunghezza asse	100
Misura angolare	Misura angolare	50
Undo/Redo	Annulla e ripeti	20
Selezione multipla	Selezione di più entità	50
Gestione Layer	Gestione dei livelli	20
Ortho	Modalità Orto	100
Proprietà	Gestione proprietà	30

Tabella 2.1: Funzionalità Cad Editor

2.4 Analisi degli ambienti CAD esistenti

Il mercato attuale offre molte soluzioni CAD, alcune libere altre a pagamento. Dopo un attenta analisi di questi prodotti, si nota che dal punto di vista delle funzionalità offerte, AutoCAD di AutoDesk è il più completo. Spesso le soluzioni software libere rappresentano un emulazione di tutto ciò che l'applicazione di AutoDesk offre. Come specificato nel Capitolo 1, Cad Editor non vuol'essere un sistema CAD a livello di AutoCAD, ma deve poter offrire una serie di strumenti limitati e allo stesso tempo sufficienti per poter creare un qualsiasi modello CAD, utile agli scopi specificati nell'introduzione.
Dall'analisi di questi ambienti CAD sono emerse le seguenti considerazioni:

- Creazione: Cad Editor non offre la possibilità di creare quote lineari e allineate. La creazione di entità quali arco, linea e cerchio sono limitate ad un unica modalità. Non esiste la possibilità di poter creare un poligono chiuso dalla polilinea.

- Manipolazione: in Cad Editor non sono implementate funzioni di manipolazione molto utili, ad esempio:

  - Offset: utile a creare linee parallele e cerchi concentrici;
  - Mirror: utile a specchiare un insieme di entità;
  - Raccordo: arrotonda un angolo tra 2 linee con un arco.

  Durante la rotazione e lo spostamento delle entità non viene visualizzata alcuna anteprima.

Queste nuove funzionalità non presenti nel software saranno implementate durante lo stage, la modalità e il funzionamento dettagliato verranno descritti nei capitoli successivi di questa tesi.

Per la completa analisi degli ambienti CAD esistenti si faccia riferimento al documento *Analisi ambienti CAD*, redatto al termine di questa analisi.

CAPITOLO 3

Analisi dei Requisiti

Il presente capitolo espone l'Analisi dei Requisiti di Cad Editor. Dapprima verrà fatta una descrizione generale presentando i casi d'uso dell'applicativo. Poi saranno esplicitati tutti i requisiti, specificandone la tipologia e descrivendoli brevemente.

3.1 Contesto d'uso e funzioni del prodotto

3.1.1 Modalità d'uso

L'applicazione Cad Editor dovrà essere utilizzata sia come plugin sia come programma a sé stante. Nel primo caso ci sarà una qualche interfaccia o pannello di collegamento tra Cad Editor e il resto dell'applicazione che lo ospita. Nel secondo caso, invece, si potrà utilizzare come un qualsiasi altro applicativo standalone. In entrambe i casi l'utente dovrà essere in grado di creare modelli CAD 2D tramite un interfaccia user-friendly. I modelli così realizzati devono poter essere salvati in formato DXF, cosicché quest'ultimi possano, in un secondo momento, venire modificati da altri sistemi CAD, come ad esempio AutoCAD.
Per creare o modificare questi modelli CAD, dovranno essere disponibili funzionalità di disegno e manipolazione entità.

3.1.2 Funzioni del prodotto

Navigazione e selezione

Per riuscire ad avere una panoramica completa di un disegno, c'è la necessità di poter effettuare uno zoom o semplicemente cambiare la porzione di modello visibile sull'area di lavoro. Nondimeno, per riferirsi in maniera univoca alle entità, c'è bisogno di un meccanismo di selezione singola e multipla. Queste 2 funzionalità devono necessariamente appartenere a Cad Editor.

Disegno di entità elementari

Rappresentano i mattoni di costruzioni che l'utente e il sistema utilizzano per creare disegni bidimensionali. Le entità elementari sono semplici geometrie quali linea, arco, circonferenza, ellisse. Viene definita elementare, anche se non propriamente geometrica, l'entità testo. Cad Editor offre la possibilità di creare tutte queste entità, ma le modalità sono molto limitate, si dovranno pensare nuove modalità per poter offrire all'utente più alternative.

Disegno di entità composte

Rappresentano strutture grafiche più complesse come poligoni, linee parallele, cerchi concentrici, raccordi e quote. Quest'ultime possono essere di diversi tipi: allineate, lineari e angolari. Queste entità mancano completamente da Cad Editor.

Punti notevoli

Fondamentali per la selezione e la modifica veloce delle entità. Inoltre, questi punti rappresentano punti cardine per la creazione di altre entità, anche non essendo delle vere e proprie entità. Si pensi ai punti notevoli di una linea: estremi e punto medio. Questi punti possono essere riferiti come centro per la creazione di una circonferenza.

Modifica del disegno

Oltre ad utilizzare i punti notevoli come rapido metodo di modifica entità, devono essere disponibili altri comandi particolari quali lo specchio di entità, in modo da velocizzare il disegno di modelli simmetrici. Le entità devono poter essere cancellate, spostate, copiate e modificate tramite le proprietà disponibili sulla griglia. Parte di queste funzionalità esistono già in Cad Editor, altre invece dovranno essere create durante lo stage.

Gestione dei livelli

Dare un ordine a disegni complessi è molto importante. Spesso accade che un modello rappresenti una sezione in cui più livelli sono visibili, e si ha la necessità di identificarli in modo diverso. L'applicazione deve poter dare la possibilità di

creare, cancellare e modificare nome e colore di questi livelli. Per ora la gestione dei livelli è solo parzialmente implementata, dovrà essere corretta e integrata.

Inserimento da tastiera

Alcune operazioni richiedono molta precisione numerica sia essa una misura o un punto. In queste situazioni l'utilizzo del solo mouse può creare problemi, è necessario poter dare all'utente la possibilità di inserimento di questi valori da tastiera. Esiste una maschera in Cad Editor che permette questa operazione, ma dovrà essere sostituita, come definito nel Piano di Lavoro.

3.1.3 Piattaforma di esecuzione

Essendo realizzato con la tecnologia .NET, Cad Editor può essere eseguito solo su sistemi operativi Windows XP o superiori, che abbiano installato almeno il Framework .NET 2.0.

3.1.4 Caratteristiche degli utenti

Per poter utilizzare questo ambiente, all'utente non sono richieste particolari conoscenze. Certamente avere familiarità con l'interfaccia ribbon (ad esempio con il pacchetto Office di Microsoft a partire dal 2007) e con applicazioni CAD, facilita notevolmente l'apprendimento. Infatti, la maggior parte delle funzionalità, avrà un metodo di utilizzo simile al popolare AutoCAD, in modo da non sconvolgere le routine che molti utenti possono aver memorizzato.

3.2 Vincoli generali

Il software dovrà:

- Essere utilizzabile su qualsiasi computer sufficientemente potente con sistema operativo Windows;
- permettere la creazione di modelli CAD 2D;
- permettere l'esportazione dei modelli in diversi formati;
- essere accompagnato da una guida utente in formato .chm;

3.3 Assunzioni e dipendenze

Cad Editor sarà progettato basandosi sulle specifiche fornite da:

- **AD01**: l'azienda IT+Robotics all'interno del Piano di Lavoro;
- **AD02**: l'azienda IT+Robotics durante lo svolgimento dello stage;
- **AD03**: auto imposti, dopo l'analisi del codice esistente di Cad Editor (Sez. 2.3);
- **AD04**: auto imposti, dopo l'analisi degli ambienti CAD presenti nel mercato (Sez. 2.4);

Gli identificativi in grassetto verranno ripresi successivamente, quando saranno elencati tutti i requisiti.

3.4 Use case

3.4.1 Use case generale - UC01

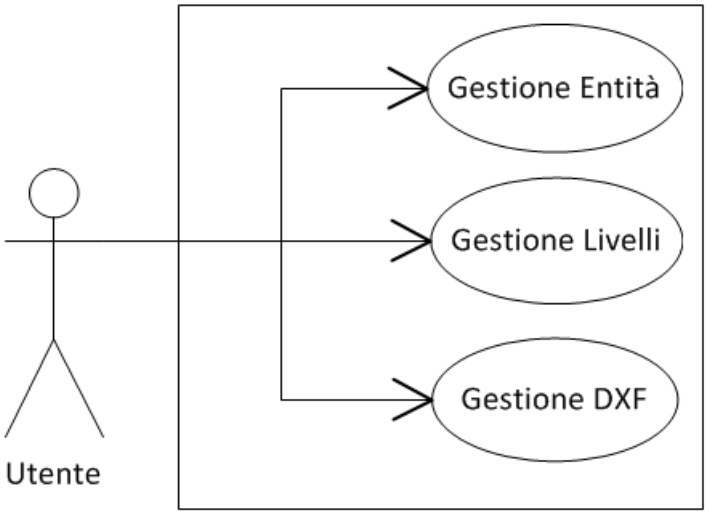

Figura 3.1: Use Case: funzionalità generali.

Attore principale: utente.

Pre-condizioni: l'utente ha aperto il sistema Cad Editor e si trova sulla schermata principale.

Breve descrizione: Cad Editor permette la creazione di modelli CAD 2D grazie alla gestione di entità (creazione e modifica) e alla gestione dei livelli di visualizzazione. Tale modello può essere salvato in formato DXF per essere caricato in un secondo momento.

Post-condizioni: il sistema ha soddisfatto le richieste dell'utente.

3.4.2 Gestione entità - UC01.1

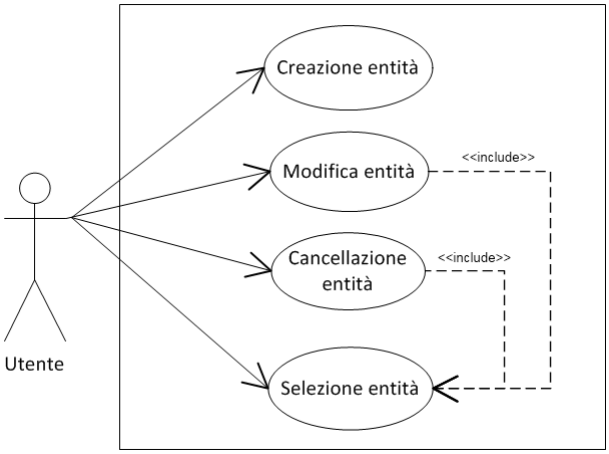

Figura 3.2: Use Case: Gestione entità.

Attore principale: utente.

Pre-condizioni: l'utente si trova sulla schermata principale. Il sistema è pronto per accettare le richieste di gestione delle entità.

Breve descrizione: quando il modello è in fase di creazione e contiene al suo interno delle entità, l'utene ha la possibilità di modificarle o di crearne di nuove. Per effettuare una modifica o una cancellazione le entità devono prima essere selezionate. La creazione dell'entità è spiegata in dettaglio nella sezione 3.4.3.

Post-condizioni: il sistema ha soddisfatto le richieste dell'utente.

3.4.2.1 Creazione entità - UC01.1.1

Per la descrizione dettagliata si veda la sezione 3.4.3.

3.4.2.2 Selezione entità

Pre-condizioni: l'utente si trova sulla schermata principale. Il sistema è pronto per accettare il click di selezione delle entità.

Funzionamento dettagliato: quando un utente ha bisogno mi modificare o cancellare una entità la deve poter selezionare. Esistono 2 modi per fare questo:

1. fare click vicino all'entità che si vuole selezionare (compaiono i marker);
2. utilizzare lo strumento di selezione multipla agendo sull'apposito pulsante. Creare l'area di selezione indicandone 2 vertici opposti.

Post-condizioni: il sistema ha selezionato le entità scelte dall'utente.

3.4.2.3 Cancellazione entità

Pre-condizioni: l'utente si trova sulla schermata principale e ha selezionato le entità che intende cancellare:

Funzionamento dettagliato: all'utente è sufficiente premere il tasto CANC e le entità selezionate saranno eliminate. E' possibile annullare l'operazione agendo sul'apposito pulsante di UNDO.

Post-condizioni: Cad Editor ha eliminato le entità dal modello.

3.4.2.4 Modifica entità - UC01.1.2

Per la descrizione dettagliata si veda la sezione 3.4.4.

3.4.3 Creazione entità - UC01.1.1

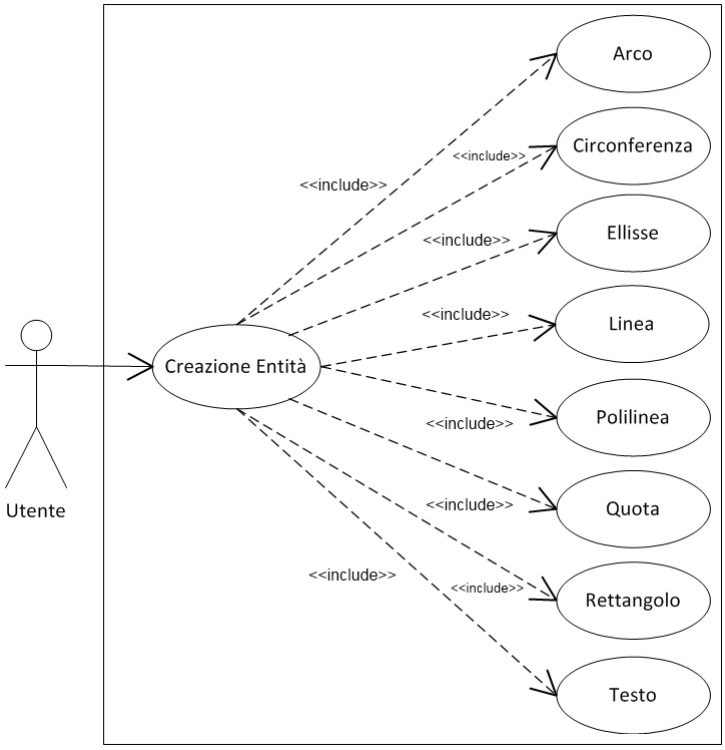

Figura 3.3: Use Case: Creazione entità.

Attore principale: utente.

Pre-condizioni: l'utente ha cliccato sulla creazione di una delle entità ed è pronto ad interagire con il View Control.

Descrizione: Cad Editor consente di disegnare diverse tipologia di entità, agendo dapprima sul pulsante che identifica l'entità da creare e successivamente interagendo sull'area di lavoro. Ogni entità ha una sequenza di azioni diversa da eseguire, sulla base della loro natura geometrica.

Funzionamento dettagliato: una volta scelta l'entità l'utente deve posizionarsi sulla porzione del View Control dove vuole venga posizionata l'entità (magari aiutandosi con i marker), e inizia a disegnarla. Ogni entità viene creata diversamente ma generalmente le azioni da compiere sono:

- posizionamento del punto d'origine dell'entità (centro di un cerchio, punto iniziale si una linea ecc..) cliccando col sinistro sul View Control oppure utilizzando l'Entity Control (maschera che sostituirà il DrawPanel, si veda sezione 2.3.5);
- modifica delle caratteristiche peculiari dell'entità agendo come al punto precedente.

L'utente può scegliere una delle seguenti entità:

- Arco;
- Circonferenza;
- Ellisse;
- Linea;
- Polilinea;
- Quota;
- Rettangolo;
- Testo;

Post-condizioni: Cad Editor ha creato l'entità con le proprietà scelte dall'utente.

3.4.4 Modifica entità - UC01.1.2

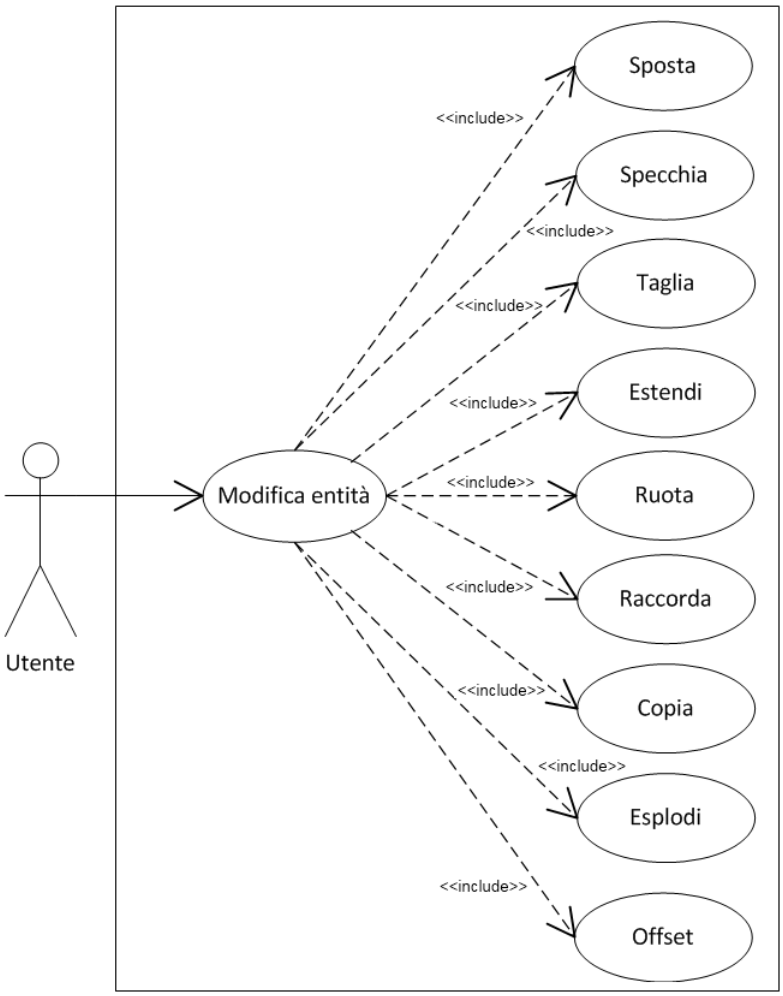

Figura 3.4: Use Case: Modifica entità.

Attore principale: utente.

Pre-condizioni: l'utente ha selezionato l'entità che intende modificare.

Descrizione: Cad Editor consente di manipolare le entità create in modo da correggere o semplicemente ampliare il modello 2D che si sta creando. Alcune funzioni di modifica semplicemente cambiano le proprietà di una certa entità, altre invece ne creano di nuove basandosi sulle proprietà delle entità selezionate (offset e mirror).

Post-condizioni: Cad Editor ha modificato l'entità come indicato dall'utente.

3.4.4.1 Sposta

Pre-condizioni: l'utente ha selezionato l'entità che intende spostare.

Descrizione: una volta creata un entità è possibile spostarla all'interno dell'area di lavoro.

Funzionamento dettagliato: per spostare un'entità l'utente può operare in 2 modi:

1. agisce sul marker di spostamento dell'entità. Cliccandoci sopra, attiva automaticamente la funzione di spostamento. Ora non gli resta che posizionarsi dove desidera e ripremere il tasto sinistro del mouse;

2. clicca sul pulsante di spostamento. Una volta attivata la funzione dovrà cliccare su un punto del View Control che rappresenterà l'origine della linea di spostamento. Infine cliccherà sul punto finale.

Post-condizioni: Cad Editor ha spostato l'entità come indicato dall'utente.

3.4.4.2 Specchia

Pre-condizioni: l'utente ha selezionato l'entità che intende spostare.

Descrizione: specchiare un'entità significa crearne una copia speculare. La posizione dello specchio sarà decisa dall'utente, ed in base ad essa la nuova entità avrà una posizione e una rotazione diversa.

Funzionamento dettagliato: per effettuare la specchiatura, l'utente deve premere sul pulsante specchia e poi deve:

1. cliccare sul punto che sarà l'origine della linea che rappresenterà lo specchio;

2. spostarsi sulla posizione dove vuole ci sia il punto terminale delo specchio;

3. cliccare in tale punto.

Post-condizioni: Cad Editor ha specchiato l'entità come indicato dall'utente.

3.4.4.3 Taglia

Pre-condizioni: l'utente si trova sulla schermata principale. Il sistema è pronto per accettare l'avvio della funzionalità Trim.

Descrizione: una linea può essere tagliata esclusivamente se una seconda linea la interseca. In questo modo si indica quale sia la linea limite e la porzione di segmento da eliminare.

Funzionamento dettagliato: per effettuare il taglio, l'utente deve premere sul pulsante Trim e poi deve:

1. selezionare la linea che intende tagliare;
2. selezionare la linea limite;
3. selezionare la porzione di linea che intende eliminare.

Post-condizioni: Cad Editor ha tagliato il segmento indicato dall'utente.

3.4.4.4 Estendi

Pre-condizioni: l'utente si trova sulla schermata principale. Il sistema è pronto per accettare l'avvio della funzionalità Extend.

Descrizione: una linea può essere estesa esclusivamente se esiste una seconda linea tale che interseca la retta su cui è posta la prima. In questo modo si indica quale sia la linea limite e la linea che si vuole estendere sino a tale limite.

Funzionamento dettagliato: per effettuare il taglio, l'utente deve premere sul pulsante Trim e poi deve:

1. selezionare la linea che intende estendere;
2. selezionare la linea limite;

Post-condizioni: Cad Editor ha esteso il segmento indicato dall'utente.

3.4.4.5 Ruota

Pre-condizioni: l'utente ha selezionato l'entità che intende ruotare.

Descrizione: una volta creata un'entità è possibile ruotarle all'interno dell'area di lavoro.

Funzionamento dettagliato: per ruotare un'entità l'utente deve:

1. clicca sul pulsante di ruota;
2. cliccare su un punto del View Control che rappresenterà l'origine della linea di rotazione;
3. cliccherà sul punto finale.

In questo modo l'entità viene ruotata e spostata a una distanza pari al segmento.

Post-condizioni: Cad Editor ha ruotato l'entità come indicato dall'utente.

3.4.4.6 Sposta

Pre-condizioni: l'utente ha selezionato l'entità che intende copiare.

Descrizione: una volta creata un'entità è possibile copiarla all'interno dell'area di lavoro.

Funzionamento dettagliato: per copiare un'entità l'utente deve:

1. clicca sul pulsante di copia;
2. cliccare su un punto del View Control che rappresenterà l'origine della linea di copia;
3. cliccherà sul punto finale.

In questo modo la copia dell'entità viene posta a una distanza pari al segmento creato.

Post-condizioni: Cad Editor ha copiato l'entità come indicato dall'utente.

3.4.4.7 Esplodi

Pre-condizioni: l'utente ha selezionato la polilinea che intende esplodere.

Descrizione: una volta creata, una polilinea può essere divisa in tutte le linee che la compongono.

Funzionamento dettagliato: per esplodere una polilinea, l'utente deve semplicemente cliccare sul pulsante di esplosione.

Post-condizioni: Cad Editor ha esploso la polilinea come indicato dall'utente.

3.4.4.8 Raccorda

Pre-condizioni: l'utente si trova sulla schermata principale.

Descrizione: quando si intendono creare angoli arrotondati, diventa oneroso farlo tramite la creazione manuale di archi posti ai limiti di 2 linee che si intersecano. A questo scopo ci viene in aiuto il comando raccorda, che indicate le 2 linee da raccordare crea automaticamente un arco ai limiti di quest'ultime eliminando o allungando la porzione di linea fino a raggiungere la lunghezza adatta.

Funzionamento dettagliato: per raccordare 2 linee l'utente deve:

1. clicca sul pulsante di raccordo;
2. indicare, tramite l'Entity Control, il raggio dell'arco di raccordo;
3. selezionare le 2 linee da raccordare;
4. indicare la posizione del centro dell'arco.

Post-condizioni: Cad Editor ha creato il raccordo tra le linee come indicato dall'utente.

3.4.4.9 Offset

Pre-condizioni: l'utente si trova sulla schermata principale.

Descrizione: spesso capita che ci sia la necessità di avere linee parallele o cerchi

concentrici. La funzione offset permette di farlo semplicemente indicando l'entità e la distanza della linea parallela o del cerchio concentrico. Questa funzionalità è una delle più utilizzate professionalmente, poiché velocizza molto la creazione di modelli 2D.

Funzionamento dettagliato: per effettuare l'offset di un'entità l'utente deve:

1. clicca sul pulsante di offset;
2. indicare, tramite l'Entity Control, la distanza;
3. selezionare l'entità di riferimento;
4. indicare la posizione di offset.

Le entità che supportano questa funzionalità sono: linea, cerchio, polilinea, ellisse e arco.

Post-condizioni: Cad Editor ha creato la nuova entità offset come indicato dall'utente.

3.4.5 Gestione dei livelli - UC01.2

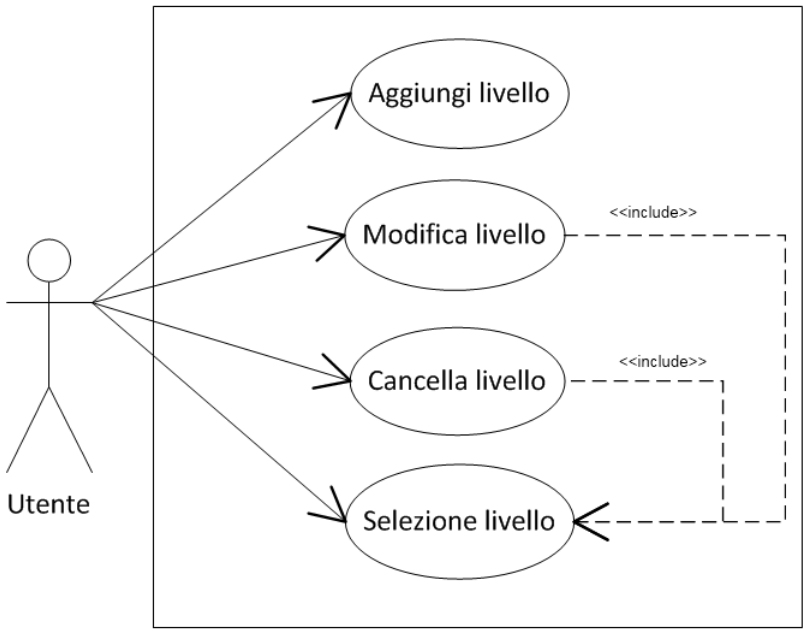

Figura 3.5: Use Case: Gestione dei livelli.

Attore principale: utente.

Pre-condizioni: l'utente si trova sulla schermata principale. Il sistema è pronto per accettare le richieste di gestione dei livelli.

Breve descrizione: per creare modelli complessi, è indispensabile avere la possibilità di ordinare le varie entità su diversi livelli, ognuno avente caratteristiche diverse. Cad Editor offre una gestione completa dei livelli che vanno dalla creazione alla modifica delle proprietà e infine alla loro cancellazione. Il pannello per fare queste operazioni è posto sulla destra del View Control.

Post-condizioni: il sistema ha soddisfatto le richieste dell'utente.

3.4.5.1 Selezione livello

Pre-condizioni: l'utente si trova sulla schermata principale. Il sistema è pronto per accettare le richieste di gestione dei livelli.

Funzionamento dettagliato: per selezionare un livello è sufficiente agire sulla grid e cliccare sulla riga corrispondente. Il pannello sottostante prenderà le caratteristiche del livello selezionato in modo che possano essere modificate.

Post-condizioni: il sistema ha selezionato il livello come richiesto dall'utente.

3.4.5.2 Aggiunta livello

Pre-condizioni: l'utente si trova sulla schermata principale. Il sistema è pronto per accettare le richieste di aggiunta di un livelli.

Descrizione: se il livello di default non è sufficiente per un'adeguata gestione del modello, l'applicazione da la possibilità di aggiungerne di nuovi.

Funzionamento dettagliato: per aggiungere un livello basta cliccare sul pulsante apposito di aggiunta livello. La griglia aggiungerà così un nuovo livello con un certo nome di default che l'utente può modificare.

Post-condizioni: il sistema ha aggiunto il livello come richiesto dall'utente.

3.4.5.3 Cancella livello

Pre-condizioni: l'utente ha selezionato il livello che vuole cancellare.

Descrizione: se un livello diviene inutile, c'è la possibilità di cancellarlo dalla lista.

Funzionamento dettagliato: per cancellare un livello basta cliccare sul pulsante apposito di cancellazione livello.

Post-condizioni: il sistema ha cancellato il livello come richiesto dall'utente.

3.4.5.4 Modifica livello

Pre-condizioni: l'utente ha selezionato il livello che vuole modificare.

Descrizione: ogni livello differisce dagli altri in base alle sue proprietà. Tipicamente ogni livello appare con un colore differente, così da facilitarne la localizza-

zione.

Funzionamento dettagliato: esistono 4 proprietà modificabili in un livello:

- Nome: semplicemente si digita il nome desiderato sulla textbox;
- Colore: si seleziona il colore grazie all'utilizzo del colorpicker;
- Attivo: è un flag che indica se il livello attivo è quello selezionato;
- Visibile: flag che indica se il livello è visibile nel modello.

Post-condizioni: il sistema ha modificato il livello come richiesto dall'utente.

3.4.6 Gestione DXF - UC01.3

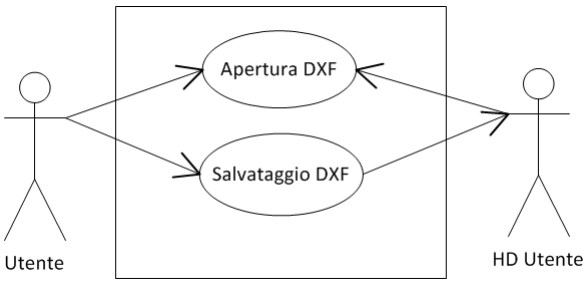

Figura 3.6: Use Case: Gestione DXF.

Attore principale: utente.
Attore secondario: HD utente.

Pre-condizioni: l'utente si trova sulla schermata principale. Il sistema è pronto per accettare le richieste di apertura/salvataggio di file DXF.

Breve descrizione: un modello 2D generalmente viene creato in più sessioni, ed è quindi utile poterlo salvare per poi caricarlo in un secondo momento. Cad Editor permette entrambe le operazioni, ed essendo il formato DXF il più utilizzato nella creazione di modelli CAD, è possibile caricare anche modelli creati con altri strumenti.

Post-condizioni: il sistema ha soddisfatto le richieste dell'utente.

3.4.6.1 Salvataggio DXF

Pre-condizioni: l'utente ha creato un modello o parte di esso e intende salvare il lavoro.

Funzionamento dettagliato: per salvare il modello in un file .dxf l'utente deve semplicemente cliccare sull'apposito pulsante di salvataggio e indicare il nome e la destinazione del file.

Post-condizioni: il sistema ha salvato il modello come richiesto dall'utente.

3.4.6.2 Apertura DXF

Pre-condizioni: l'utente ha creato un modello o parte di esso e intende salvare il lavoro.

Funzionamento dettagliato: per salvare il modello in un file .dxf l'utente deve semplicemente cliccare sull'apposito pulsante di salvataggio e indicare il nome e la destinazione del file.

Post-condizioni: il sistema ha salvato il modello come richiesto dall'utente.

3.4.7 Interfaccia iniziale - UC02

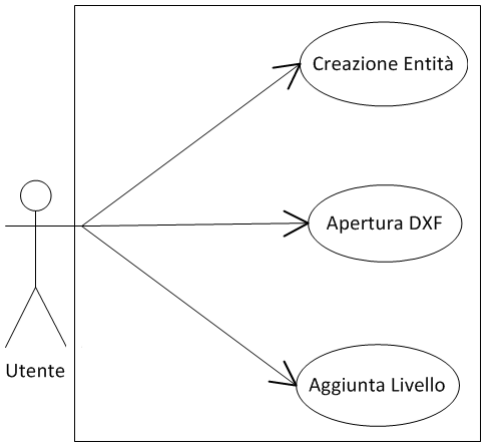

Figura 3.7: Use Case: Interfaccia iniziale.

Attore principale: utente.

Pre-condizioni: l'utente ha aperto il sistema Cad Editor e si trova sulla schermata principale.

Descrizione: appena avviato, il programma, visualizza l'interfaccia grafica (Sez. 2.3.1), e si trova da subito in modalità disegno. E' possibile quindi iniziare il disegno grazie agli appositi comandi, oppure si può aprire un modello DXF precedentemente salvato (anche in un altro ambiente). C'è anche la possibilità di aggiungere livelli, nonostante il modello non contenga alcuna entità.

Funzionamento dettagliato: l'utente appena avviato il programma può:

1. cliccare su uno dei pulsanti di creazione entità, e quindi essere pronto per disegnare una specifica entità nel modello;
2. cliccare sul pulsante di apertura DXF, per caricare un modello precedentemente salvato;
3. aggiungere dei livelli agendo sul pannello dei livelli posto sulla destra dell'interfaccia.

Per una descrizione dettagliata di queste funzionalità si vedano rispettivamente le sezioni 3.4.3, 3.4.6.2, 3.4.5.2.

Post-condizioni: in base all'operazione effettuata dall'utente il sistema di trova in 3 stati diversi:

1. il sistema è in attesa che l'utente interagisca con il View Control per disegnare l'entità selezionata;
2. il sistema ha caricato il file DXF ed è pronto a ricevere istruzioni;
3. il sistema ha effettuato le modifiche ai livelli come richiesto dall'utente.

3.4.8 Misura - UC03

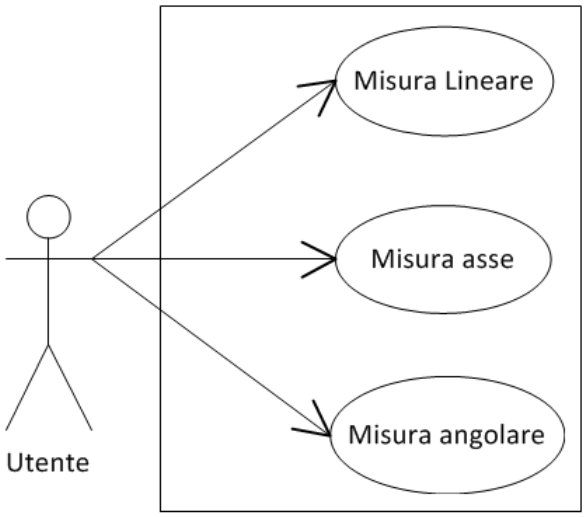

Figura 3.8: Use Case: Misura.

Attore principale: utente.

Pre-condizioni: l'utente si trova sulla schermata principale.

Breve descrizione: può essere utile durante la costruzione di un modello CAD effettuare delle misure sia per questioni di controllo, sia per avere una panoramica delle dimensioni del disegno. Cad Editor offre 3 diversi tipi di misurazione:

- Misura lineare: ritorna la distanza tra 2 punti;
- Misura asse: ritorna la distanza tra 2 punti in base a un asse;
- Misura angolare: ritorna la misura dell'angolo tra 2 linee.

Post-condizioni: il sistema ha soddisfatto le richieste dell'utente.

3.4.8.1 Misura lineare

Pre-condizioni: l'utente ha creato un modello o parte di esso e intende effettuare una misura.

Funzionamento dettagliato: per fare una misura l'utente deve:

1. cliccare sul pulsante di misura;
2. cliccare sul primo punto;
3. cliccare sul secondo punto.

Post-condizioni: il sistema ha ritornato la misura sulla status bar.

3.4.8.2 Misura asse

Pre-condizioni: l'utente ha creato un modello o parte di esso e intende effettuare una misura di un asse.

Funzionamento dettagliato: per fare una misura l'utente deve:

1. cliccare sul pulsante di misura asse, x o y;
2. cliccare sul primo punto;
3. cliccare sul secondo punto.

Post-condizioni: il sistema ha ritornato la misura asse sulla status bar.

3.4.8.3 Misura angolare

Pre-condizioni: l'utente ha creato un modello o parte di esso e intende effettuare una misura angolare.

Funzionamento dettagliato: per fare una misura l'utente deve:

1. cliccare sul pulsante di misura angolare;
2. cliccare sulla prima linea;
3. cliccare sulla seconda linea.

Post-condizioni: il sistema ha ritornato la misura angolare sulla status bar.

3.5 Lista dei requisiti

Tutti i requisiti sotto indicati seguiranno la seguente catalogazione:

XYZ: [descrizione del requisito] (AD, Use-Case)

- **X**: Indica il tipo di requisito, che può essere
  - **F** : requisito Funzionale
  - **R** : requisito Prestazionale
  - **Q** : requisito di Qualità
  - **V** : requisito di Vincolo
- **Y**: Rilevanza strategica dei requisiti
  - **B** : requisito Obbligatorio
  - **P** : requisito Opzionale
  - **D** : requisito Desiderabile
- **Z** : Numero progressivo a due cifre che identifica il requisito (01, 02, ...n);
- **AD** : Indica l'Assunzione o la Dipendenza da cui deriva il requisito (Sez. 3.3);
- **Use-Case**: Indica il nome dell'eventuale Use-Case associato a quel requisito.

Si specifica che, nel caso di requisiti gerarchici, si terrà conto della seguente notazione:

XYZ.X'Y'Z'

dove XYZ indica il codice del requisito padre e X'Y'Z' il codice del requisito figlio, sempre seguendo le descrizioni sopracitate. Z rappresenta il contatore incrementale rispetto al proprio livello gerarchico.

3.5.1 Requisiti Funzionali

Requisiti obbligatori

- *FB01 - Gestione Entità*: permettere all'utente di gestire le entità che comporranno il modello CAD 2D. (AD01, UC01.1);
  - *FB01.FB01 - Aggiunta entità*: mettere a disposizione dell'utente le seguenti entità:
    - Arco;
    - Circonferenza;
    - Ellisse;
    - Linea;
    - Polilinea;
    - Rettangolo;
    - Quota;
    - Testo;

    (AD01, UC01.1.1)
  - *FB01.FB02 - Cancellazione entità*: permettere all'utente di eliminare un'entità precedentemente creata. (AD01, UC01.1);
  - *FB01.FB03 - Modifica entità*: offrire all'utente i seguenti strumenti di manipolazione delle entità:
    - Sposta: muove un'entità;
    - Ruota: ruota un'entità;
    - Specchia: crea una copia specchiata dell'entità;
    - Taglia: taglia una linea sulla base di una seconda linea limita;
    - Estendi: allunga una linea sulla base di una seconda linea limita;
    - Raccorda: crea un raccordo (arco) nell'angolo formato da 2 linee non parallele tra loro;
    - Copia: crea una copia di un'entità;
    - Esplodi: divide una polilinea nelle linee che la compongono;
    - Offset: crea curve concentriche e linee parallele.

    (AD01 e AD02, UC01.1.2)

  - *FB01.FB04 - Selezione entità*: gestire la selezione singola e multipla delle entità. (AD01 e AD02, UC01.1.2);

- *FB02 - Gestione View Control*: gestire le azioni dell'utente che avvengono sull'area di lavoro (movimenti e click del mouse). (AD01);

- *FB03 - Gestione Entity Control*: gestire una nuova maschera di inserimento valori da tastiera, che comparirà in basso a destra dell'interfaccia. (AD01);

- *FB04 - Gestione dei livelli*: permettere all'utente di creare livelli custom modificandone le proprietà. (AD01 e AD02, UC01.2);

  - *FB04.FB01 - Creazione livelli*: permettere all'utente di creare nuovi livelli per la gestione del proprio modello CAD. (AD02, UC01.2);
  - *FB04.FB02 - Cancellazione livelli*: permettere all'utente di eliminare dei livelli precedentemente creati. (AD02, UC01.2);
  - *FB04.FB03 - Modifica livelli*: dare la possibilità all'utente di modificare le proprietà dei livelli, che sono: colore, nome, visibilità e attività. (AD02, UC01.2);

- *FB05 - Gestione file DXF*: permettere l'apertura e il salvataggio di modelli in formato DXF. (AD01, UC01.3);

  - *FB05.FB01 - Apertura file DXF*: permettere l'apertura di file formato DXF, creato anche con software diverso da Cad Editor. (AD01, UC01.3);
  - *FB05.FB02 - Salvataggio file DXF*: permettere il salvataggio del modello in formato DXF. (AD01, UC01.3);

- *FB06 - Misurazioni*: permettere all'utente di effettuare delle misure. (AD01, UC03);

  - *FB06.FB01 - Misura angolare*: permettere all'utente di effettuare una misura angolare tra 2 linee. (AD01, UC03);
  - *FB06.FB02 - Misura lineare*: permettere all'utente di effettuare una misura lineare tra 2 punti. (AD01, UC03);

  - *FB06.FB02 - Misura lineare*: permettere all'utente di effettuare una misura dell'asse (x o y) tra 2 punti. (AD01, UC03);

- *FB07 - Gestione dei punti di interesse*: gestire i punti di interesse (marker) delle entità, in modo da velocizzare l'operazione di modifica e per dare un segno distintivo alle entità selezionate. (AD01);

- *FB08 - Griglia Proprietà*: visualizzare e permettere la modifica delle proprietà delle entità selezionate (se più di una le proprietà visualizzate sono la loro intersezione) tramite griglia. (AD01 e AD02);

- *FB09 - Esportazione immagine*: permettere l'esportazione del modello in vari formati immagine (jpg, gif, png, svg). (AD01);

- *FB10 - Cronologia*: dare la possibilità all'utente di annullare e/o ripetere un operazione. (AD01).

3.5.2 Requisiti prestazionali

Requisiti desiderabili

- *RD01 - Prestazione*: il software deve essere il più leggero possibile e l'utilizzo deve essere il più fluido possibile. Il View Control deve rispondere bene ai comandi dell'utente senza ritardi o errori grafici.

3.5.3 Requisiti di qualità

Requisiti obbligatori

- *QB01 - Guida utente*: creare una guida utente in formato .chm. (AD02);

- *QB02 - Guida tecnica*: allegare al prodotto software una guida tecnica riguardo alla progettazione dell'applicazione, per agevolare eventuali espansioni di Cad Editor. (AD02);

Requisiti desiderabili

- *QD01 - OOP*: il codice deve, per quanto possibile, essere scritto seguendo il paradigma della programmazione orientata agli oggetti. (AD04);

Requisiti opzionali

- *QP01 - Commento del codice*: codice commentato e diviso in regioni (comando #region del C#). (AD02);
- *QP02 - Pulizia del codice*: il codice deve essere il più semplice e pulito possibile. (AD02);

3.5.4 Requisiti di vincolo

Requisiti obbligatori

- *VB01 - Tecnologie*: realizzazione del software utilizzando il linguaggio C# in ambiente Visual Studio 2008 di Microsoft;
- *VB02 - Piattaforma*: il prodotto software deve funzionare correttamente su sistemi operativi Windows XP o successivi con installato almeno il Framework .NET 2.0.

Requisiti desiderabili

- *VD01 - Grafica*: il software deve poter funzionare correttamente anche su schede grafiche non particolarmente potenti.

Capitolo 4
Progettazione

In questo capitolo verranno esposte le scelte progettuali adottate per la definizione del software Cad Editor. Si inizierà con una descrizione del funzionamento generale, fino ad arrivare alla spiegazione in dettaglio delle varie componenti.

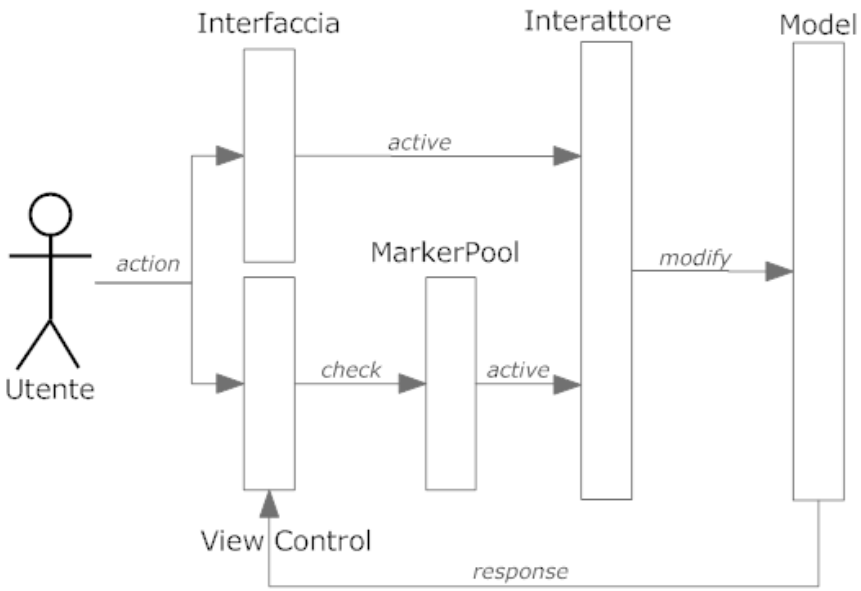

Figura 4.1: Funzionamento ad alto livello.

La figura 4.1 mostra, ad alto livello, quello che accade quando un utente effettua una certa operazione. Sostanzialmente l'utente può agire in 2 diversi modi: utilizzare uno degli strumenti presenti all'interno della ribbon bar, oppure interagire con l'area di lavoro (ViewControl) ed effettuare una modifica manuale. Entrambe le modalità attivano un interattore, con l'unica differenza che non è il ViewControl ad attivarlo direttamente, ma passa attraverso la classe MarkerPool (questo aspetto sarà esaminato in dettaglio in una sezione successiva). Infine, tramite l'interattore, il modello viene modificato e questa modifica si ripercuote nel ViewControl che si

aggiorna. Questo è ciò che accade generalmente, la natura molto varia delle funzioni porta in alcuni casi a meccanismi leggermente differenti.

Come detto nella sezione 2.3.6, la struttura o la forma delle classi dell'applicazione era insoddisfacente, per questo motivo è stato svolto un refactoring iniziale che ha cambiato radicalmente l'architettura del software. Definita tale nuova struttura sono state implementate le funzionalità come descritto nel *Piano di Lavoro*. Le sezioni che seguono descrivono sia le parti del software che sono state riviste, sia le parti create. Tutto ciò che non ha riguardato lo stage non viene trattato all'interno di questa sezione, e in generale all'interno di questa tesi.

4.1 Specifica delle Componenti

4.1.1 Gerarchia Cad Objects

Per iniziare, è stata definita una gerarchia di classi che rappresentano le entità. Lo schema delle classi di questa gerarchia lo si può vedere in figura 4.2.

Una classe di tipo CadObject costituisce un involucro (wrapper) per gli oggetti di tipo DxfEntity che sono definiti nella libreria CadLib. La scelta di creare un wrapper deriva dal fatto che le entità così come sono in CadLib sono molto limitate (Sez. 2.2.6). Le proprietà che le contraddistinguono sono minime e richiedono molti calcoli per essere aggiornate in maniera dinamica durante il disegno. Una classe CadObject quindi cerca di colmare queste lacune aggiungendo nuove proprietà e funzioni.
Le manipolazioni di queste proprietà vengono eseguite dai metodi di tali classi, risolvendo così il problema delle funzioni generiche, presentato nella sezione 2.3.6.
Anche la definizione dei Marker e l'attivazione degli interattori vengono effettuate da queste classi. Prima questi compiti venivano svolti dalla classe MarkerPool che, come detto nella sezione 2.3.4, era un contenitore di MarkerInteractor (da ora più presenti). Si ha quindi che un oggetto di tipo Cad Object rappresenta in tutto e per tutto un'entità CAD sia per caratteristiche che per funzionalità ad essa applicabili. Ogni entità conosce i propri punti di interesse e sa cosa accade agendo su ognuno di essi. L'interazione in sé, viene delegata a un DrawInteractor, che resta comunque strettamente legato all'entità a cui fa riferimento.

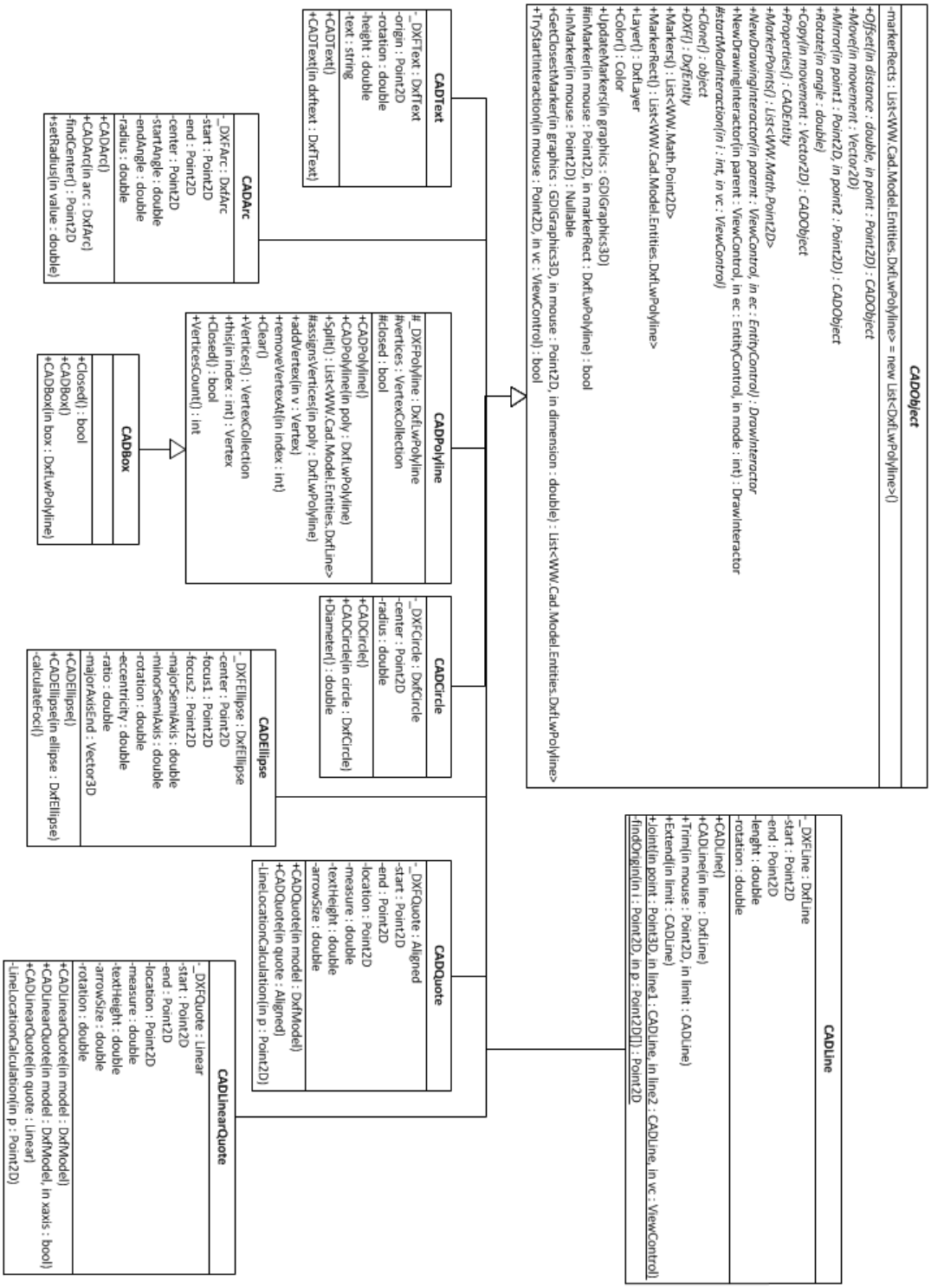

Figura 4.2: Gerarchia Cad Objects.

4.1.1.1 CADObject

La classe CADObject è la classe base della gerarchia, da cui tutte le altre classi, direttamente o transitivamente, derivano. É una classe astratta e quindi non può essere istanziata, questo perché non ha alcun senso creare un'entità generica priva di forma. La classe implementa l'interfaccia IClonable così da poter definire il metodo *Clone* utile alla copia degli oggetti. Al suo interno è presente un unico campo dati, denominato *markerRects*, che è un vettore di marker grafici che tutte le entità devono possedere per poter visualizzare i propri punti notevoli.

Relazioni da e verso altre componenti

CadObject è utilizzato all'interno del ViewControl, poiché quest'ultimo si occupa di salvare e visualizzare il modello. Il modello che viene visualizzato accetta come entità esclusivamente DxfEntity e non CADObject, è necessario quindi che esista un doppio riferimento tra questi tipi di oggetti. A questo scopo all'interno del ViewControl esiste un vettore associativo (Dizionario) che lega oggetti di tipo DxfEntity a oggetti CADObject i quali al loro interno contengono un riferimento al DxfEntity corrispondente.

Di seguito sono elencati i metodi astratti (indicati in corsivo) e non che sono presenti in questa classe, con una breve descrizione. Saranno discussi più approfonditamente nelle classi derivate se di particolare rilevanza.

Metodi

- *public CADObject Offset(double, Point2D)*
  Metodo che effettua l'offset dell'entità in cui è definito. Il parametro double indica la distanza dell'offset mentre il punto, che rappresenta la posizione del mouse, è utile a capire la direzione in cui effettuare l'offset. Viene restituito un nuovo CADObject perché l'offset non manipola l'entità ma ne crea una nuova sulla base della stessa.

- *public void Move(Vector2D)*
  Metodo che esegue lo spostamento dell'entità in base al vettore che gli viene passato come parametro. Il vettore è bidimensionale poiché non serve la coordinata z.

- *public void Rotate(double)*
  Metodo che ruota l'entità sulla base dell'angolo che gli viene passato come riferimento. L'ampiezza dell'angolo è riferita a un vettore ideale posto sopra l'asse x e avente la stessa direzione.

- *public CADObject Copy(Vector2D)*
  Sostanzialmente uguale al metodo Move con la differenza che al posto di spostare l'entità ne crea una copia e la sposta.

- *public CADEntity Properties()*
  Questo metodo non riguarda la manipolazione dell'entità, ma restituisce le sue proprietà visibili nella griglia. L'oggetto di tipo CADEntity generico deve essere specificato per ogni entità e al suo interno sono racchiusi metodi get-set (chiamati appunto proprietà nel linguaggio C#) delle proprietà della singola entità.

- *object Clone()*
  Metodo derivato dall'interfaccia IClonable che serve per creare copie profonde di oggetti di tipo CADObject.

- *public CADObject Mirror(Point2D, Point2D)*
  Metodo che specchia l'entità. I due parametri rappresentano i punti che definiscono la linea di specchio. La nuova entità specchiata sarà speculare a quella di origine in base allo specchio. La Figura 4.3 chiarisce il concetto.

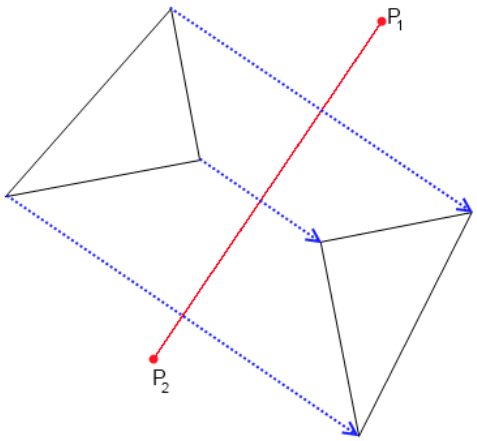

Figura 4.3: Funzionamento del metodo Mirror.

I punti P_1 e P_2 sono i due parametri che definiscono lo specchio rosso. Le frecce indicano l'isomorfismo tra punti attraverso lo specchio. Il parametro restituito costituisce l'entità specchiata.

- *public List¡Point2D¿ MarkerPoints()*
  Metodo che restituisce i punti notevoli dell'entità tramite una lista di Point2D.

- *public DxfEntity DXF*
  Proprietà che restituisce o imposta l'oggetto di tipo DxfEntity (entità in CadLib). Da notare che in caso l'oggetto di tipo DXF contenuto in un CADObject viene modificato anche le proprietà conseguentemente vengono aggiornate in modo da delegare qualsiasi tipo di controllo alla classe. Questo vale per questa proprietà e per tutte le proprietà definite nelle varie entità.

- *public List¡DxfLwPolyline¿ MarkerRect*
  Proprietà che si limita a restituire il campo dati markerRects. DxfLwPolyline è il tipo polilinea che rappresenta un rettangolo a livello grafico.

- public DxfLayer Layer
  Proprietà che restituisce o imposta il layer dell'entità. DxfLayer è il layer in CadLib che di default è posto a 0.

- public System.Color Color
  Proprietà che restituisce o imposta il colore dell'entità.

- public void UpdateMarkers(GDIGraphics)
  Metodo che aggiorna i marker a livello grafico. Questo metodo è utile quando si effettuano zoom all'interno dell'area di lavoro. Se non venissero aggiornati i marker zoomando il rettangolo che li contraddistingue si ingrandirebbe e diventerebbe graficamente incomprensibile. Effettuando l'aggiornamento invece, la grandezza rimane sempre la medesima.

- public Nullable InMarker(Point2D)
  Metodo pubblico che controlla se un certo punto è all'interno del rettangolo che rappresenta il marker.

- protected bool InMarker(Point2D, DxfLwPolyline)
  Metodo protetto che restituisce se un certo punto è all'interno di uno specifico rettangolo che rappresenta un marker.

- public Point2D GetClosestMarker(GDIGraphics, Point2D)
  Metodo che restituisce il marker più vicino alla posizione del mouse rappresentata dal Point2D passato come parametro alla funzione.

- *public DrawInteractor NewDrawingInteractor(ViewControl, EntityControl, int)*
  Metodo che restituisce un nuovo interattore per l'entità. Alcune entità possono venire create in diversi modi, questo viene definito dall'intero. Il parametro EntityControl serve per definire il tipo di maschera da visualizzare durante l'interazione. Sarà dedicata una sezione all'EntityControl che verrà spiegato dettagliatamente.

- public DrawInteractor NewDrawingInteractor(ViewControl, EntityControl)
  Metodo che restituisce un nuovo interattore di default per l'entità. Questo metodo è definito virtuale e non astratto perché può non essere ridefinito se l'entità ha un unica modalità di creazione.

- *protected void startModInteractor(int, ViewControl)*
  Metodo che gestisce l'attivazione degli interattori in base al marker selezionato. Ora i Marker Interactor non esistono più, tutto viene gestito da questa gerarchia. L'intero indica il numero del marker che l'utente ha scelto.

- public bool TryStartInteraction(Point2D, ViewControl)
  Metodo che controlla se è stata attivata un interazione. Viene testato il punto in cui si trova il mouse rispetto ai marker, e se si trova all'interno di uno di questi parte l'interazione.

4.1.1.2 CADArc

Classe che rappresenta un arco. Un arco può essere definito come parte di una circonferenza. I punti di interesse di un arco sono: il centro della circonferenza di cui è parte, il punto di inizio, il punto di fine e il punto medio dell'arco. Rispettivamente questi marker modificano, la posizione, il punto di inizio, il punto di fine, e il raggio dell'arco.

Campi dati

- DxfArc _DXFArc: rappresenta l'oggetto di tipo DxfCircle che è racchiuso dal CADArc;
- Point2D start: rappresenta il punto di inizio dell'arco;
- double startAngle: rappresenta l'angolo di inizio dell'arco;
- Point2D end: rappresenta il punto di fine dell'arco;
- double endAngle: rappresenta l'angolo di fine dell'arco;
- Point2D center: rappresenta il centro della circonferenza a cui appartiene l'arco.
- double radius: rappresenta il raggio della circonferenza a cui appartiene l'arco.

Metodi

- private Point2D findCenter()
  Metodo che trova il centro in caso siano dati solo i 2 punti di inizio, fine e il raggio della circonferenza.

4.1.1.3 CADCircle

Classe che rappresenta una circonferenza. I Punti di interesse di un'entità di questo tipo sono il centro e un punto arbitrariamente scelto della circonferenza. Rispettivamente modificano la posizione e il raggio della circonferenza.

Campi dati

- DxfCircle _DXFCircle: rappresenta l'oggetto di tipo DxfCircle che è racchiuso dal CADCircle;
- Point2D center: rappresenta il centro della circonferenza.
- double radius: rappresenta il raggio della circonferenza.

Metodi

- public double Diameter()
  Metodo che restituisce il diametro ricavandolo dal doppio del raggio.

4.1.1.4 CADEllipse

Classe che rappresenta un ellisse. I Punti di interesse di un ellisse sono: centro, punto terminale asse maggiore e punto terminale asse minore. Rispettivamente modificano la posizione, il semiasse maggiore e il semiasse minore.

Campi dati

- DxfEllipse _DXFEllipse: rappresenta l'oggetto di tipo DxfEllipse che è racchiuso dal CADEllipse;
- Point2D center: rappresenta il centro dell'ellise;
- double majorSemiAxis: rappresenta il semiasse maggiore dell'ellise;
- double minorSemiAxis: rappresenta il semiasse minore dell'ellise;
- Point2D focus1, focus2: rappresentano i fuochi dell'ellisse;
- double rotation: rappresentano la rotazione dell'asse maggiore;
- Vector2D majorAxisEnd: rappresenta il vettore che parte dal centro e termina sul punto di terminazione dell'asse maggiore;
- double eccentricity, ratio: rappresentano l'eccentricità e il rapporto asse minore su asse maggiore.

Metodi

- privare void calculateFoci()
  Metodo che ricalcola la posizione dei fuochi dopo una modifica dell'ellisse.

4.1.1.5 CADLine

Classe che rappresenta una linea o più precisamente un segmento. I Punti di interesse di una linea sono: punto di inizio, punto di fine e punto medio. Rispettivamente modificano l'inizio, la fine e la posizione. Modificando inizio e fine viene modificata anche la rotazione.

Campi dati

- DxfLine _DXFLine: rappresenta l'oggetto di tipo DxfLine che è racchiuso dal CADLine;
- Point2D start: rappresenta il punto di inizio della linea;
- Point2D end: rappresenta il punto di fine della linea;
- double lenght: rappresenta la lunghezza della linea;
- double rotation: rappresenta la rotazione del segmento.

Metodi

- public void Trim(Point2D, CADLine)
  Metodo che taglia una linea. Il parametro di tipo CADLine rappresenta la linea limite che delimita il taglio. Il punto 2D invece indica la parte da eliminare scelto dall'utente tramite click del mouse.
- public void Extend(CADLine)
  Metodo che effettua l'estensione della linea fino ad una seconda linea limite indicata dal parametro CADLine.
- public void Joint(Point2D, CADLine, CADLine, ViewControl)
  Metodo statico che effettua il raccordo tra 2 linee. L'arco che rappresenta il raccordo viene automaticamente aggiunto al modello e viene aggiornato il ViewControl. Il Point2D serve per capire la direzione del raccordo, visto che tra 2 linee che si intersecano sono possibili 4 tipi di raccordo (Vedi Figura 4.4).
- private Point2D findOrigin(Point2D, Point2D[])
  Metodo di supporto per il calcolo dell'arco di raccordo.

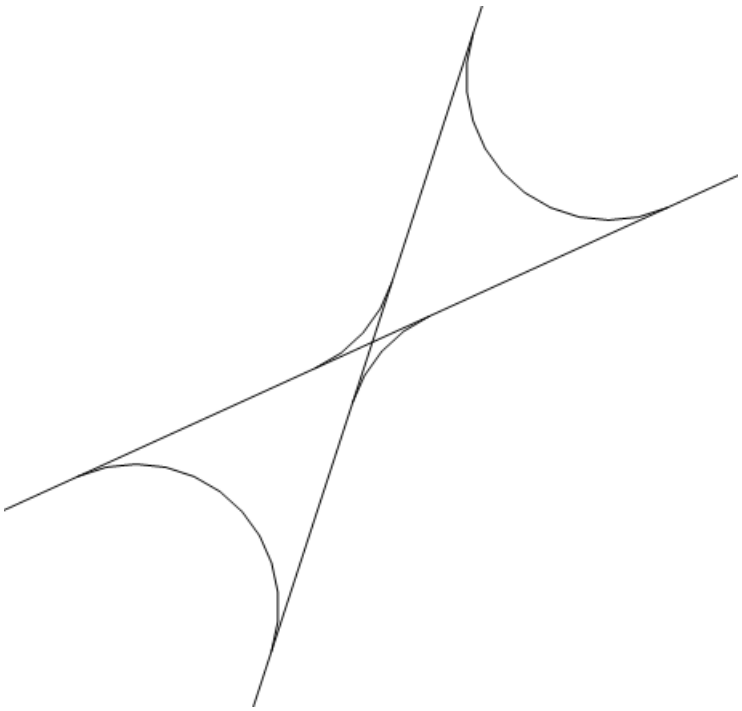

Figura 4.4: I 4 raccordi possibili di 2 linee.

4.1.1.6 CADtext

Classe che rappresenta del testo. I Punti di interesse di un testo sono: origine del testo, punto posto sull'altezza del testo e punto sulla base. Rispettivamente modificano la posizione, l'altezza e la rotazione.

Campi dati

- DxfText _DXFText: rappresenta l'oggetto di tipo DxfText che è racchiuso dal CADText;
- Point2D origin: rappresenta l'origine del testo;
- double rotation: rappresentano la rotazione del testo;
- string text: rappresenta la riga di testo;
- double height: rappresenta l'altezza del testo;

4.1.1.7 CADPolyline

Classe che rappresenta una polilinea. I Punti di interesse di una polilinea sono tutti i vertici che se modificati modificano le 2 linee che sono legate a esso.

Campi dati

- DxfLwPolyline _DXFPolyline: rappresenta l'oggetto di tipo DxfLwPolyline che è racchiuso dal CADPolyline;
- VertexCollection vertices: rappresenta la lista di vertici della polilinea;
- bool closed: booleano che indica se la polilinea rappresenta un poligono chiuso.

Metodi

- public List¡DxfLine¿ Split()
  Metodo che effettua l'esplosione della polilinea nelle linee che la compongono.

4.1.1.8 CADBox

Classe che rappresenta un rettangolo, che non è altro che un caso particolare di polilinea composta da 4 vertici e con le linee 2 a 2 perpendicolari. I punti di interesse sono i 4 vertici. Da notare che una volta disegnato un rettangolo diventa una polilinea.

Non contiene campi propri ne metodi rilevanti.

4.1.1.9 CADQuote

Classe che rappresenta una quota allineata. Innanzitutto spieghiamo la differenza tra quota lineare e allineata. La quota allineata è posizionata parallela al segmento che misura e riporta esattamente la lunghezza di tale segmento. Una quota lineare invece, è sempre parallela all'asse X o Y e misura la lunghezza dell'asse corrispondente. I punti di interesse di una quota allineata sono i punti di contatto con il segmento che misura e il punto medio della linea di quota. Rispettivamente modificano i punti di quota e l'altezza della linea di quota.

Campi dati

- DxfDimension.Aligned _DXFQuote: rappresenta l'oggetto di tipo DxfDimension.Aligned che è racchiuso dal CADQuote;
- Point2D start: rappresenta il punto iniziale di aggancio al segmento;
- Point2D end: rappresenta il punto finale di aggancio al segmento;
- Point2D location: rappresenta il punto medio della linea di quota;
- double measure: rappresenta la misura della quota;
- double textHeight, arrowSize: rappresentano l'altezza del testo e la dimensione della freccia di quota.

Metodi

- private void LineLocationCalculation(Point2D)
  Metodo che calcola la posizione della linea di quota in base alla posizione del mouse sul ViewControl.

4.1.1.10 CADLinearQuote

Classe che rappresenta una quota lineare. Del tutto simile alla precedente con la differenza di essere lineare e non allineata. Non verranno ripetuti i metodi e i campi dati che sono contenuti anche in CADQuote.

Campi dati

- DxfDimension.Linear _DXFLinearQuote: rappresenta l'oggetto di tipo DxfDimension.Linear che è racchiuso dal CADLinearQuote;
- double rotation: rappresenta la rotazione (parallela all'asse x o y).

4.1.2 Entity Control

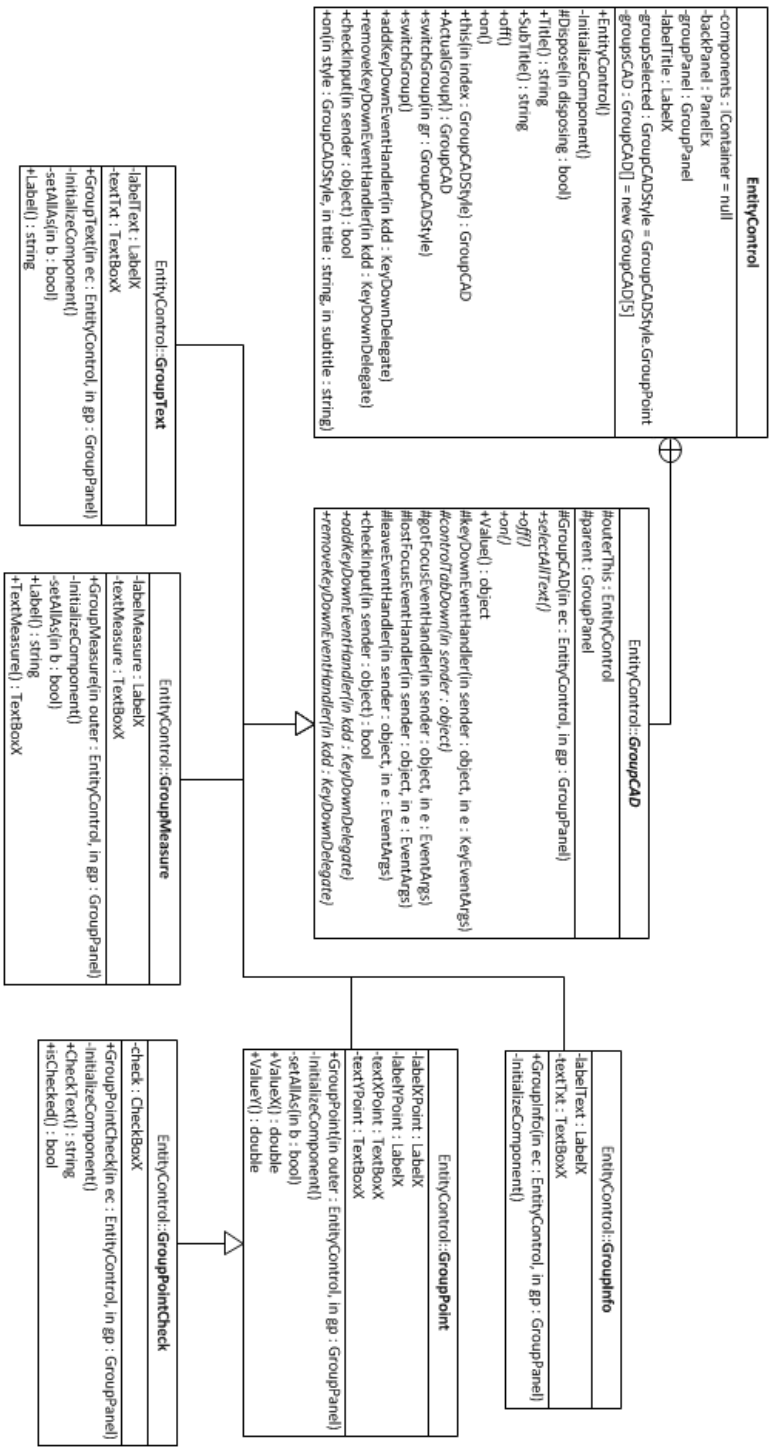

Figura 4.5: Classe Entity Control.

La figura 4.5 descrive la struttura della classe EntityControl. Questa classe rappresenta la maschera sostitutiva a DrawPanel (Sez. 2.3.5). A differenza del suo precedente, EntityControl (da qui EC) appare come un baloon sull'angolo in basso a destra dell'interfaccia di Cad Editor, senza occupare spazio alcuno all'interno del View Control. Ogni qualvolta che appare l'EC, esso mantiene il focus senza però disturbare il disegno all'interno dell'area di lavoro. EC cattura ogni pressione di tasto da parte dell'utente e effettua controlli sull'input numerico. Inoltre, si aggiorna ad ogni movimento del mouse, così da poter vedere in tempo reale le coordinate esatte del puntatore. Il suo aspetto lo si può vedere in figura 4.6. In EC

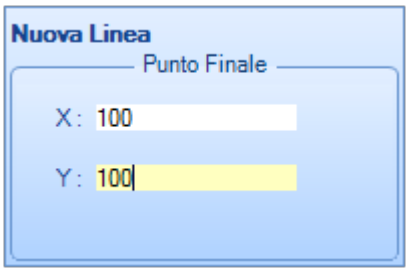

Figura 4.6: Estetica del baloon Entity Control.

possono essere inseriti diversi tipi di dati:

- Punto: genericamente sono 2 textbox che accettano valori di tipo double;
- Numerico: accetta un unico valore double;
- Testo: accetta qualsiasi tipo di input alfanumerico;
- Punto + Controllo: oltre ad avere le 2 textbox per valori double, ha una checkbox per attivare/disattivare un certo flag;
- Info: questa modalità non accetta input, ma serve semplicemente come un visore di informazioni.

La struttura della classe EC permette di creare nuove modalità di inserimento in modo molto semplice. É sufficiente estendere la classe interna GroupCAD implementandone i metodi astratti e se necessario quelli virtuali (per maggiori dettagli si veda la sezione 4.1.2.1).

Relazioni da e verso altre componenti

La classe EntityControl è riferita da tutti gli interattori, poichè oltre al mouse l'interazione da parte dell'utente può avvenire da tastiera. É inoltre riferita anche all'interno di alcuni metodi delle classi CADObject, sempre legate all'attivazione di un interattore. Al suo interno EC non contiene alcun legame con altre classi.

Di seguito sono elencati i campi e i metodi principali della classe.

Campi dati

- GroupCAD[] groupsCAD: rappresenta la lista dei diversi tipi di dati che EC permette di inserire. Allo stato attuale questo vettore sarà lungo 5.

Metodi

- public string Title
  Proprietà che serve per leggere o impostare il titolo dell'EC;
- public string SubTitle
  Proprietà che serve per leggere o impostare il sottotitolo dell'EC;
- public void on(GroupCADStyle, string, string)
  Metodo che attiva l'EC. Come parametri accetta la tipologia dell'input, e le due stringhe di titolo e sottotitolo. GroupCADStyle è un enumerazione che contiene l'elenco di tutte le tipologie esistenti di input.
- public void off()
  Metodo che disabilita l'EC, rendendolo non visibile.
- public void switchGroup(GroupCADStyle)
  Metodo che cambia la tipologia di input in base al GroupCADStyle indicato.
- public bool checkInput(object)
  Metodo che controlla l'input della textbox indicata come parametro. L'input deve essere double per essere accettato.

4.1.2.1 GroupCAD

Classe base della gerarchia di tipologie di input. Per creare un nuovo tipo di maschera (che è di tipo groupbox) si deve derivare da questa classe che offre i metodi utili per la visualizzazione e l'aggiornamento dei componenti.

Campi dati

- EntityControl outerThis: riferimento all'EC di appartenenza.

Metodi

- *public void on ()*
  Metodo che attiva la groupbox contenente le componenti della maschera.
- *public void off()*
  Metodo che disabilita la GroupCAD, rendendola non visibile.
- public object Value
  Proprietà che restituisce (è necessario un cast) o imposta il valore che l'input accetta (numerico, punto, testo ecc..).
- public bool checkInput(object)
  Metodo che controlla l'input della textbox indicata come parametro. L'input deve essere double per essere accettato.

Da questa classe base derivano le seguenti classi, ognuna indica una tipologia diversa di input:

- **GroupPoint**: accetta input a 2 double, ovvero le coordinate di 2 punti;
- **GroupMeasure**: accetta un unico double, una misura semplice;
- **GroupText**: accetta qualsiasi tipo di stringa;
- **GroupPointCheck**: deriva da GroupPoint ma aggiunge un controllo tramite checkbox;
- **GroupInfo**: non accetta alcun input, ma offre solo informazioni statiche.

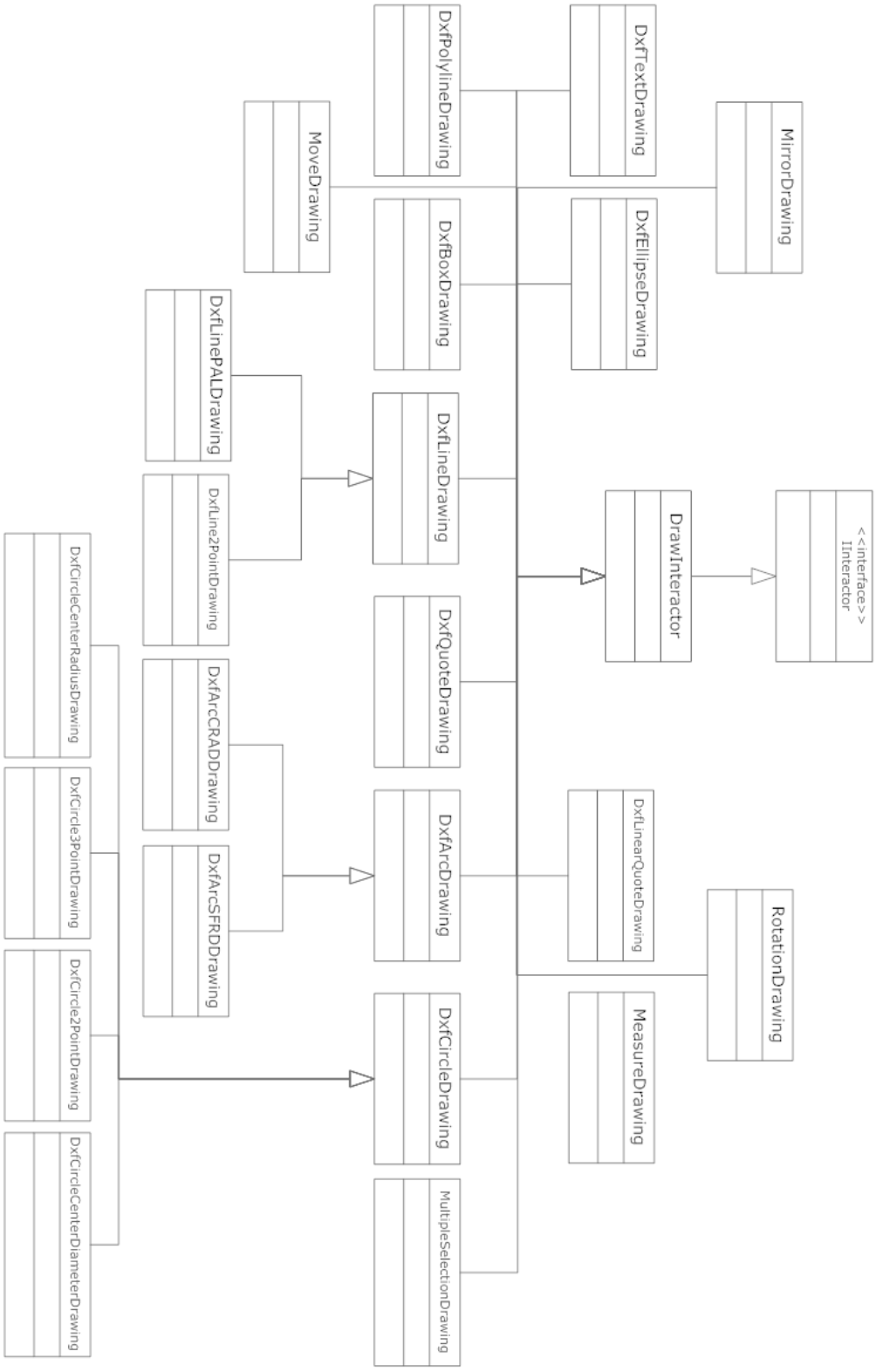

Figura 4.7: Gerarchia di classi degli interattori.

4.1.3 Interattori

La figura 4.7 rappresenta la gerarchia degli interattori. Come si può notare i MarkerInteractor non sono presenti, poiché la gestione dei marker viene eseguita direttamente dalle classi che derivano da CADObject. Un interattore per essere creato deve implementare l'interfaccia IInteractor offerta da CadLib. In questo caso questa implementazione è fatta all'interno classe base della gerarchia, ovvero DrawInteractor, che poi viene sempre derivata. La figura, per problemi di spazio, non indica metodi e campi delle diverse classi, questi saranno discussi successivamente in questa sezione.

4.1.3.1 DrawInteractor

Come detto nella sezione precedente, questa classe costituisce la base di tutta la gerarchia. L'interazione grafica-utente rappresenta il fulcro dell'applicazione, questo rende molto delicato ed importante la funzione di questo componente. Deve poter gestire tutti gli eventi inerenti a mouse e tastiera, delegando, quando necessario, la gestione della pressione dei tasti all'EntityControl. Oltre a tenere aggiornato il modello in base alle modifiche correnti che l'utente sta eseguendo, deve offrire un anteprima, senza creare errori grafici o ritardi indesiderati. A questo scopo vengono in aiuto le ottimizzazioni offerte dalla classe GDIGraphics della libreria CadLib.

Relazioni da e verso altre componenti

Le classi di questa gerarchia vengono create e avviate ogni qualvolta che un oggetto di tipo CADObject attiva un interazione. Al suo interno si riferiscono sia al ViewControl su cui avvengono le modifiche, sia all'EntityControl che permette all'utente l'inserimento dei valori da tastiera.

Metodi

- *public void ProcessMouseDown(MouseEventArgs)*
  Evento che gestisce la pressione del tasto sinistro (e destro) del mouse. Provoca l'avanzamento dello stato dell'interazione.

- *public void ProcessMouseUp(MouseEventArgs)*
  Evento che gestisce il rilascio del tasto sinistro (e destro) del mouse.

- *public void ProcessMouseWheel(MouseEventArgs)*
  Evento che gestisce la rotazione della rotellina del mouse. Di default effettua lo zoom del modello.

- *public void ProcessMouseMove(MouseEventArgs)*
  Evento che gestisce lo spostamento del mouse. Provoca l'aggiornamento dell'anteprima dell'interazione.

- *public void Activate(DxfModel, IGraphics3D)*
  Metodo che attiva l'interattore inizializzando i campi dati.

- void Deactivate()
  Metodo che disattiva l'interattore e aggiorna il modello.

- *protected void TextX_KeyDown(object, KeyEventArgs)*
  Evento che gestisce la modifica delle textbox dell'EntityControl. Cattura la pressione dei pulsanti della tastiera.

Questi metodi sono realizzati all'interno di ogni singola classe che deriva da DrawInteractor, sia essa di creazione o di manipolazione. Ognuna di queste classi è in grado di gestire sequenze di azioni diverse, data la differente natura di ogni entità.
Quando avviene un interazione di modifica tramite pressione di un marker, queste sequenze di azioni possono essere ridotte. Ad esempio se si clicca sul marker del punto finale di una linea, viene attivato l'interattore in modalità modifica, indicando che l'azione di scelta del punto iniziale è già avvenuta. In questo modo ora l'utente dovrà selezionare solo il punto finale, poi automaticamente il LineInteractor si disattiva, facendo ritornare l'applicazione nello stato normale.

Le altre classi della gerarchia non verranno discusse dettagliatamente, poiché del tutto simile alla classe DrawInteractor e quindi di poco interesse.
Interessante però è il fatto che all'interno della gerarchia esistano più interattori di alcuni tipi di entità, questo perché si è scelto di dare la possibilità all'utente di creare alcune entità con modalità diverse. Di seguito sono elencate le entità aventi più modalità di interazione:

- Arco:

1. **DxfArcCRADrawing**: disegno dell'arco dato il centro, il raggio, l'angolo di inizio e di fine;
2. **DxfArcSFRDrawing**: disegno dell'arco dato punto di inizio, punto di fine e raggio della circonferenza che lo contiene.

- Circonferenza:

  1. **DxfCircleCenterRadiusDrawing**: disegno della circonferenza dato centro e raggio;
  2. **DxfCircleCenterDiameterDrawing**: disegno della circonferenza dato centro e diametro;
  3. **DxfCircle2PointDrawing**: disegno della circonferenza dati 2 punti opposti della stessa;
  4. **DxfCircle3PointDrawing**: disegno della circonferenza dati 3 punti contenuti in essa.

- Linea:

  1. **DxfLinePALDrawing**: disegno di una linea dato punto di origine, angolo di rotazione e lunghezza;
  2. **DxfLine2PointDrawing**: disegno di una linea dati i punti di inizio e fine.

Esistono poi interattori legati ad alcune delle funzionalità di manipolazione o misura:

- **MirrorDrawing**: interattore della funzione specchia;
- **MultipleSelectionDrawing**: interattore della funzionalità di selezione multipla delle entità;
- **MoveDrawing**: interattore della funzione di spostamento;
- **RotationDrawing**: interattore della funzione di rotazione;
- **MeasureDrawing**: interattore per la funzione di misurazione.

4.1.4 MarkerPool

Dopo aver effettuato il refactoring, la classe MarkerPool ha perso molte delle funzionalità per la quale è stata creata. Infatti, al suo interno non esiste più la lista di MarkerInteractor (non esistono nemmeno i MarkerInteractor), ma ora si limita a controllare se sono avvenute interazioni con i marker, e se ciò accade attiva l'interattore corrispondente tramite il CADObject a cui si riferisce il marker.

Relazioni da e verso altre componenti

MarkerPool viene creata all'interno del ViewControl, il quale delega il controllo della pressione dei tasti del mouse o del movimento a MarkerPool. Quest'ultimo interagisce con le classi CADObject tramite il meccanismo descritto nell'introduzione di questa sezione.

Gli eventi che gestiscono il mouse vengono tralasciati poiché del tutto simili a quelli della classe DrawInteractor.

Metodi

- private List¡DxfEntity¿ GetEntitiesNearMarkerPoints(Point2D)
  Metodo che restituisce la lista delle entità vicine al marker indicato dal punto passato come parametro.

4.1.5 ViewControl

A livello generale, le funzioni svolte dalla classe ViewControl non cambiano rispetto a quelle esposte nella sezione 2.3.2, cambia però il modo in cui queste vengono svolte. Di seguito (Figura 4.8) si può vedere il diagramma di collaborazione corrispondente a quello di figura 2.2. Come si può notare facilmente, la classe CADLine, della gerarchia CADObject, assume un ruolo fondamentale, mentre viene ridotto il lavoro svolto dalla classe MarkerPool. Concettualmente tutto ciò che avviene è analogo allo schema precedente: appena l'utente clicca sul pulsante di aggiunta linea, avviene la creazione di una CADLine, il quale si occupa a sua volta della creazione dell'interattore corrispondente. CADLine inoltre, attiva l'EntityControl che comparirà sulla GUI in basso a destra. Successivamente viene allertato il ViewControl dell'inizio dell'interazione. Fin qui l'unica differenza è l'oggetto CADLine

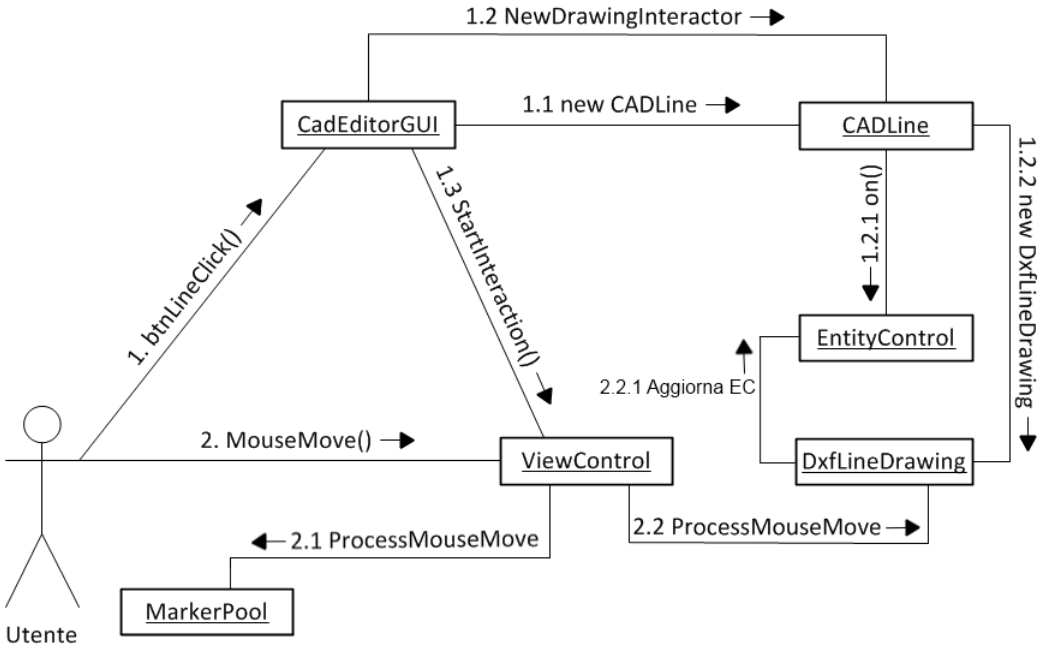

Figura 4.8: Diagramma di collaborazione del ViewControl.

che prima non esisteva. Ora l'utente muovendo il mouse all'interno dell'area di lavoro, attiva l'evento MouseMove del ViewControl, il quale chiede il controllo dei marker alla classe MarkerPool, mentre a DxfLineDrawing viene lasciato il compito di gestire l'interazione vera e propria, aggiornando l'EntityControl.
I ruoli delle varie classi ora sono meglio definiti:

- ViewControl: si occupa dell'aspetto dell'area di lavoro e di mantenere aggiornato il modello, facendo da tramite per quanto riguarda le azioni dell'utente;
- MarkerPool: si occupa della visualizzazione e del controllo dei marker;
- CADLine: e più in generale un qualsiasi oggetto della gerarchia CADObject, si occupa di mantenere aggiornata una singola entità, definendone le proprietà e i marker. Di quest'ultimi viene anche gestita l'interazione.
- DxfLineDrawing: e più in generale un qualsiasi oggetto della gerarchia degli interattoro, si occupa dell'interazione in se, seguendo l'utente in tutte le azioni che deve svolgere per completare la creazione o la modifica dell'entità.

Relazioni da e verso altre componenti

Un'istanza di ViewControl si trova all'interno della classe CadEditorGUI, questo perché l'area di lavoro è parte integrante dell'aspetto grafico dell'applicazione. ViewControl, come si è potuto vedere anche nel diagramma di collaborazione, dialoga con il MarkerPool e contiene al suo interno il vettore associativo DxfEntity-CADObject.

Di seguito verranno elencati i campi dati e i metodi più rilevanti della classe ViewControl.

Campi dati

- Dictionary¡DxfEntity, CADObject¿ modelObjects: è il vettore associativo, che associa a una data DxfEntity il suo CADObject corrispondente che a sua volta contiene il DxfEntity;
- DxfModel model: rappresenta il modello vero e proprio che viene poi disegnato tramite il GDIGraphics3D;
- GDIGraphics3D gdiGraphics3D: è l'oggetto che rende possibile la renderizzazione 2D del modello;
- DrawInteractor interactor: rappresenta l'interattore attivo. Quando nessun interattore è attivo l'oggetto è null.

Metodi

- public void StartInteraction(IInteractor)
  Metodo che inizializza l'interazione, mentre questa rimane attiva il ViewControl delegherà ogni azione all'interattore.
- private void interactor_Deactivated(object, EventArgs)
  Metodo che disattiva l'interazione portando ViewControl allo stato normale.
- protected void OnMouseWheel(MouseEventArgs)
  Evento che gestisce la rotazione della rotellina del mouse. Di default effettua lo zoom in o lo zoom out del modello.

- protected void OnMouseMove(MouseEventArgs)
  Evento che gestisce lo spostamento del mouse;
- protected void OnMouseDown(MouseEventArgs)
  Evento che gestisce la pressione del tasto sinistro del mouse;

4.1.6 Resoconto modifiche

Nella tabella che segue sono indicate, tramite segno di spunta, quali funzionalità, che erano già implementate, sono state riviste o completamente reimplementate.

Funzione	Rivisto
Creazione	
Arco	✓
Cerchio	-
Rettangolo	✓
Ellisse	✓
Linea	-
Polilinea	✓
Testo	✓
Manipolazione	
Trim	✓
Extend	-
Move	✓
Copy	✓
Rotate	✓
IO	
Export Img	-
Export PDF	-
Export DXF	-
Stampa	-
Open DXF	✓
Extrude	-
Altre	
Misura lineare	✓
Misura asse	✓
Misura angolare	✓
Undo/Redo	✓
Selezione multipla	✓
Gestione Layer	✓
Ortho	✓
Proprietà	✓

Tabella 4.1: Funzionalità Cad Editor riviste

4.2 Diagrammi di attività

Nella figura 4.9 è descritto il funzionamento del metodo OnMouseDown della classe ViewControl, che viene chiamato ogni qualvolta viene eseguito un click sull'area di lavoro. Il principale compito di questo metodo è di controllare se il click effettuato dall'utente ha un effetto di tipo interattivo, di selezione o di modifica. Il primo controllo vede se sono attive funzionalità di modifica quali offset o joint, in tal caso viene eseguito lo step corrispondente. Altrimenti, se è attivo il MarkerPool avviene la selezione, ovvero l'attivazione dei marker, dell'entità più vicina alla posizione del mouse. Infine, se è in atto un interazione questa procede secondo il proprio funzionamento.

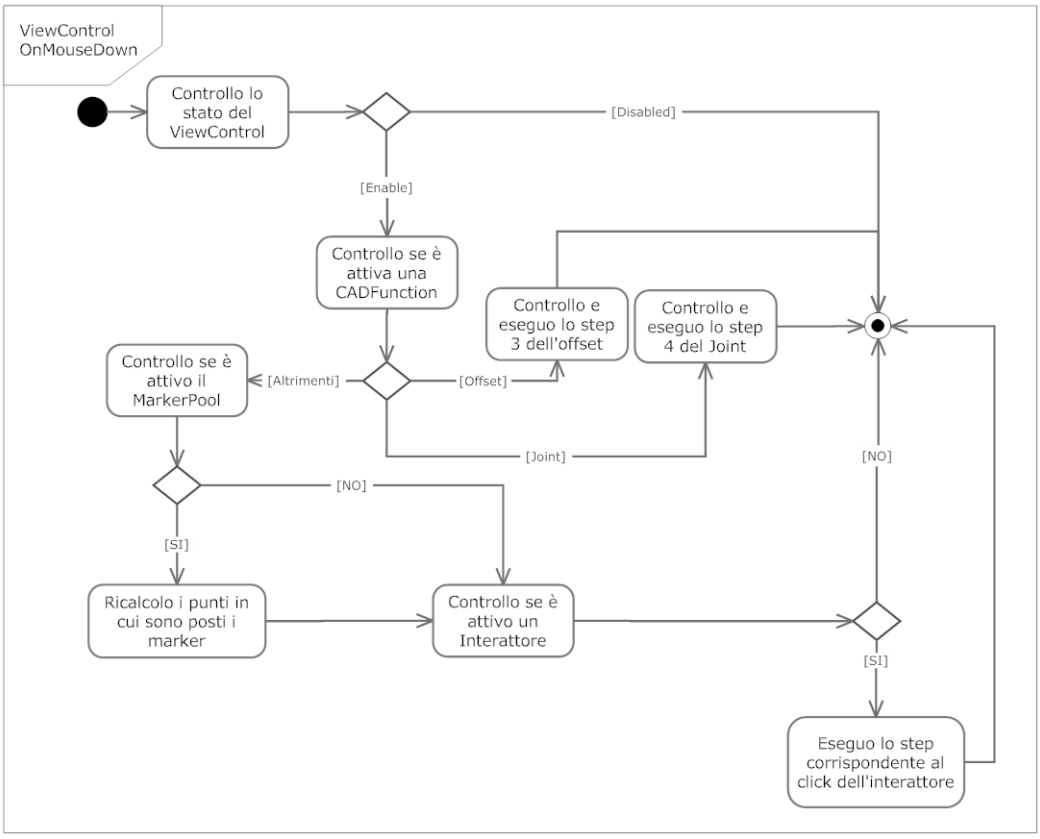

Figura 4.9: Diagramma di attività: metodo OnMouseDown della classe ViewControl.

In figura 4.10 è descritto il funzionamento del metodo StartInteraction della classe ViewControl, che viene chiamato ogni volta che un interazione inizia. Il metodo come prima cosa controlla se è in corso un'interazione. Se attiva, la termina per poter rendere possibile l'attivazione della nuova e deseleziona tutte le entità.

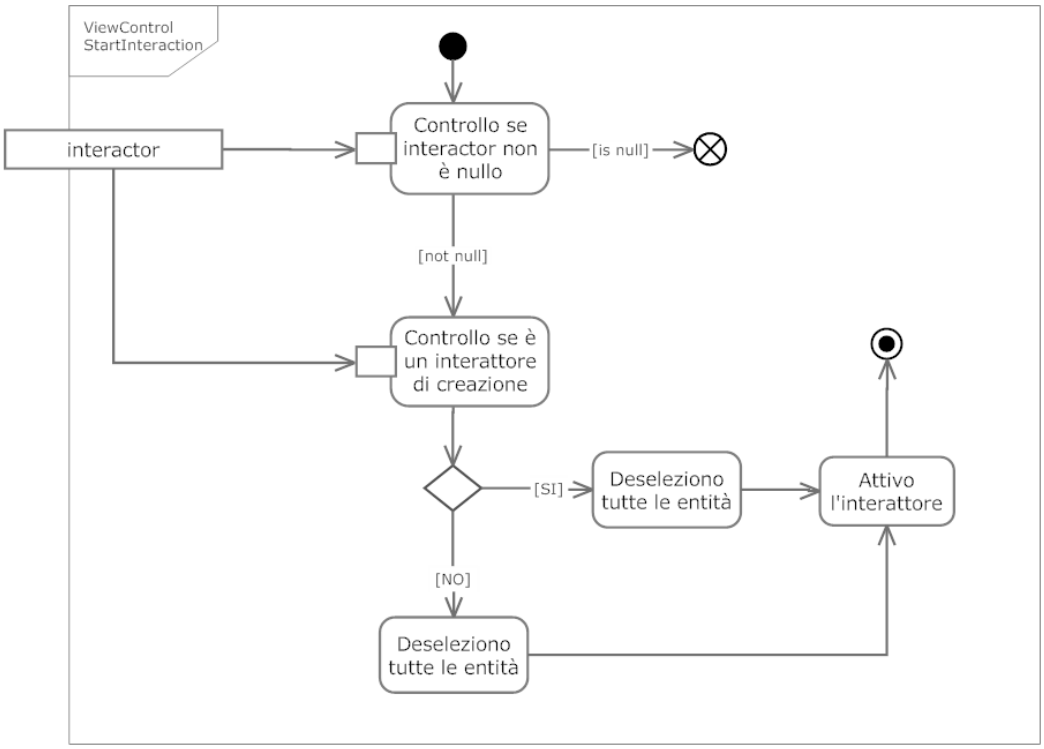

Figura 4.10: Diagramma di attività: metodo StartInteraction della classe ViewControl.

La figura 4.11 descrive il funzionamento del metodo SelectedEntity della classe CadEditorGUI, il quale viene chiamato ogni volta che un'entità viene selezionata. Da notare che un'entità che è selezionata può comunque essere selezionata, ma tale selezione non ha alcun effetto. Una volta effettuata la selezione, il metodo vede se e quale funzionalità CAD è attiva, in caso lo sia esegue lo step corrispondente.

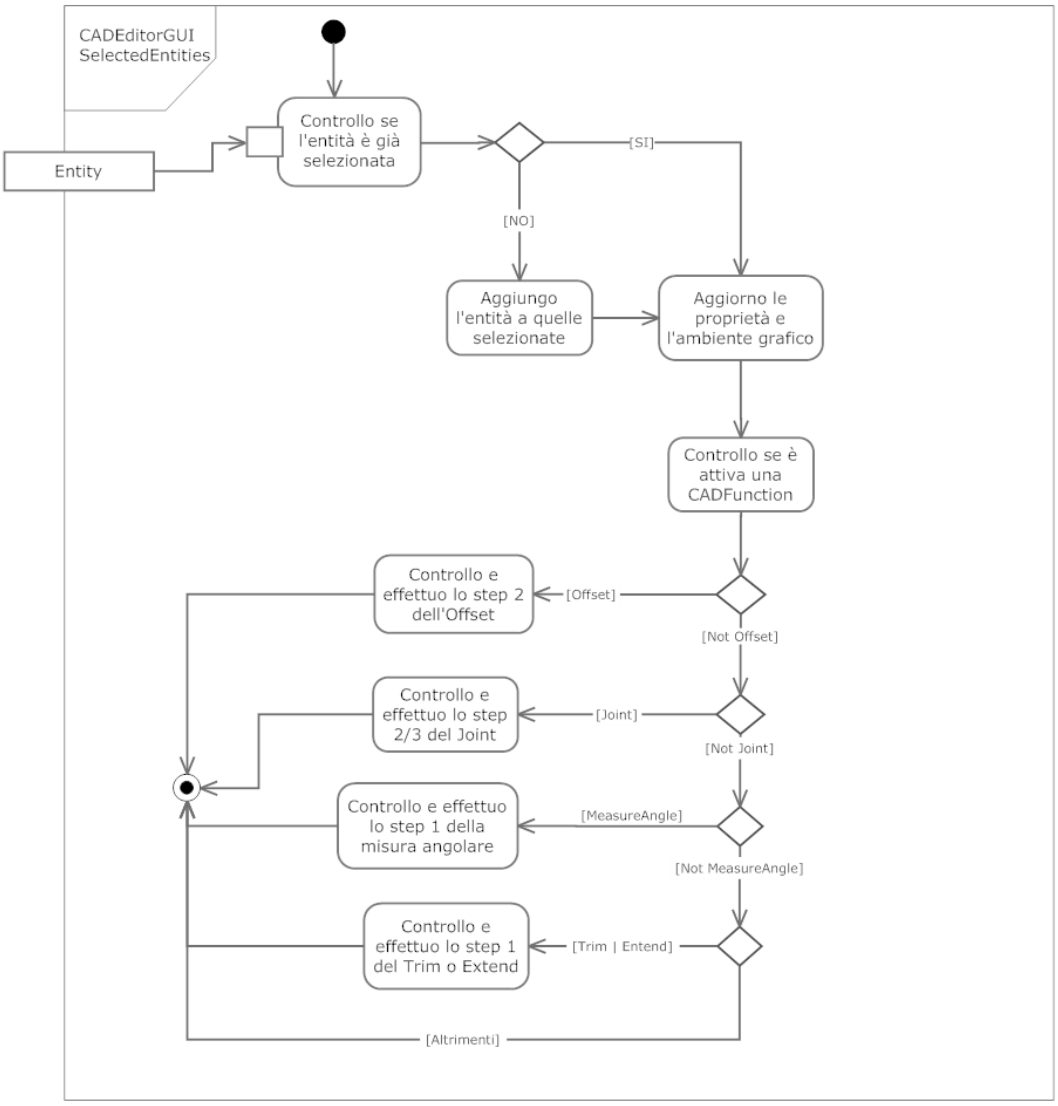

Figura 4.11: Diagramma di attività: metodo SelectedEntity della classe CadEditorGUI.

4.3 Diagrammi di sequenza

Nella figura 4.12 si descrive la sequenza di chiamate di funzione che avvengono durante l'apertura di un file DXF, eventualmente creato anche con un software diverso da CadEditor. L'utente per avviare l'operazione deve aver selezionato il file.

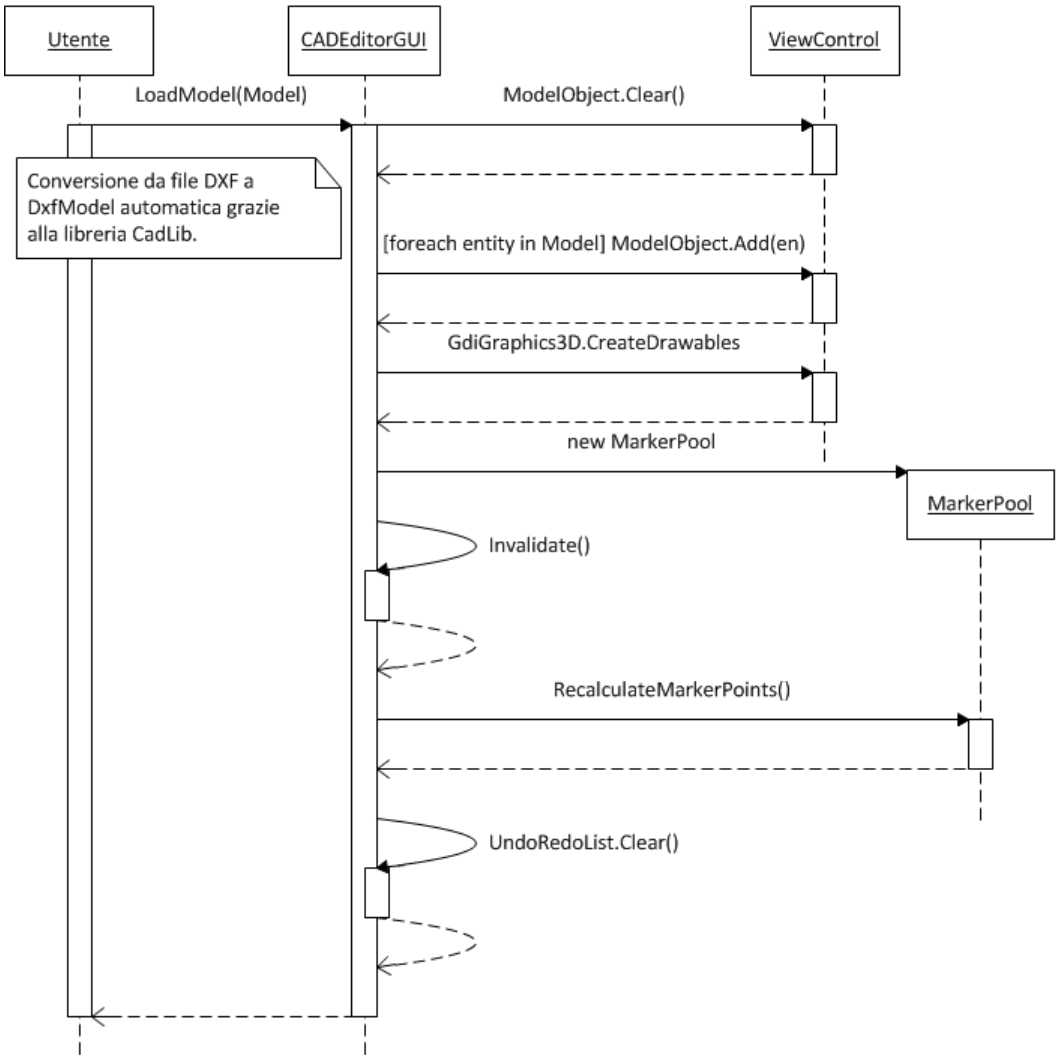

Figura 4.12: Diagramma di sequenza: caricamento file DXF.

Nella figura 4.13 si descrive la sequenza di chiamate di funzione che avvengono durante la creazione di una linea. Si noti che l'utente deve effettuare i 2 click per indicare l'inizio e la fine della linea.

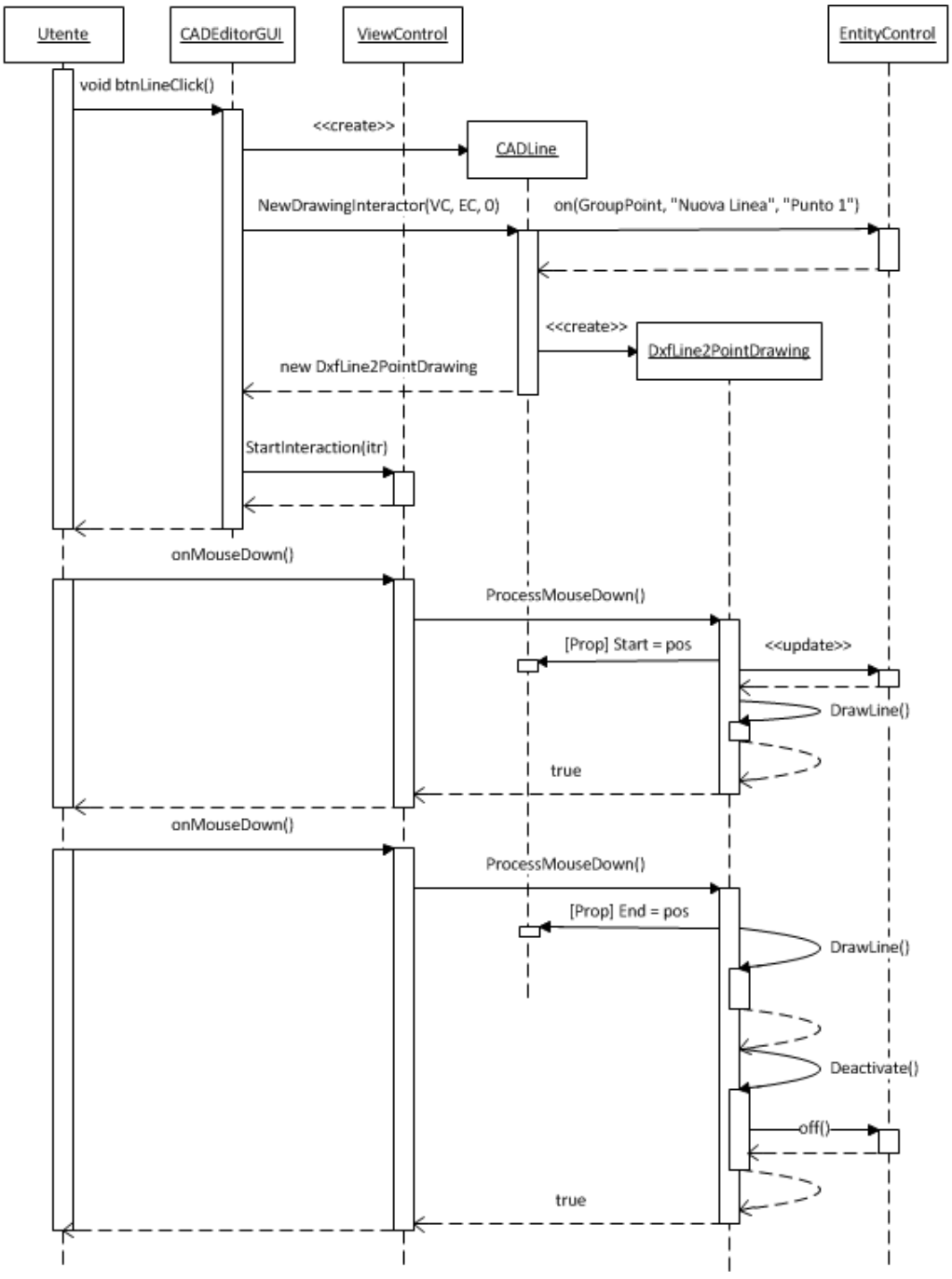

Figura 4.13: Diagramma di sequenza: creazione linea.

Nella figura 4.14 si descrive la sequenza di chiamate di funzione che avvengono durante l'offset di una linea. Si noti che l'utente deve effettuare i 2 click di selezione e l'inserimento da tastiera della dimensione di offset.

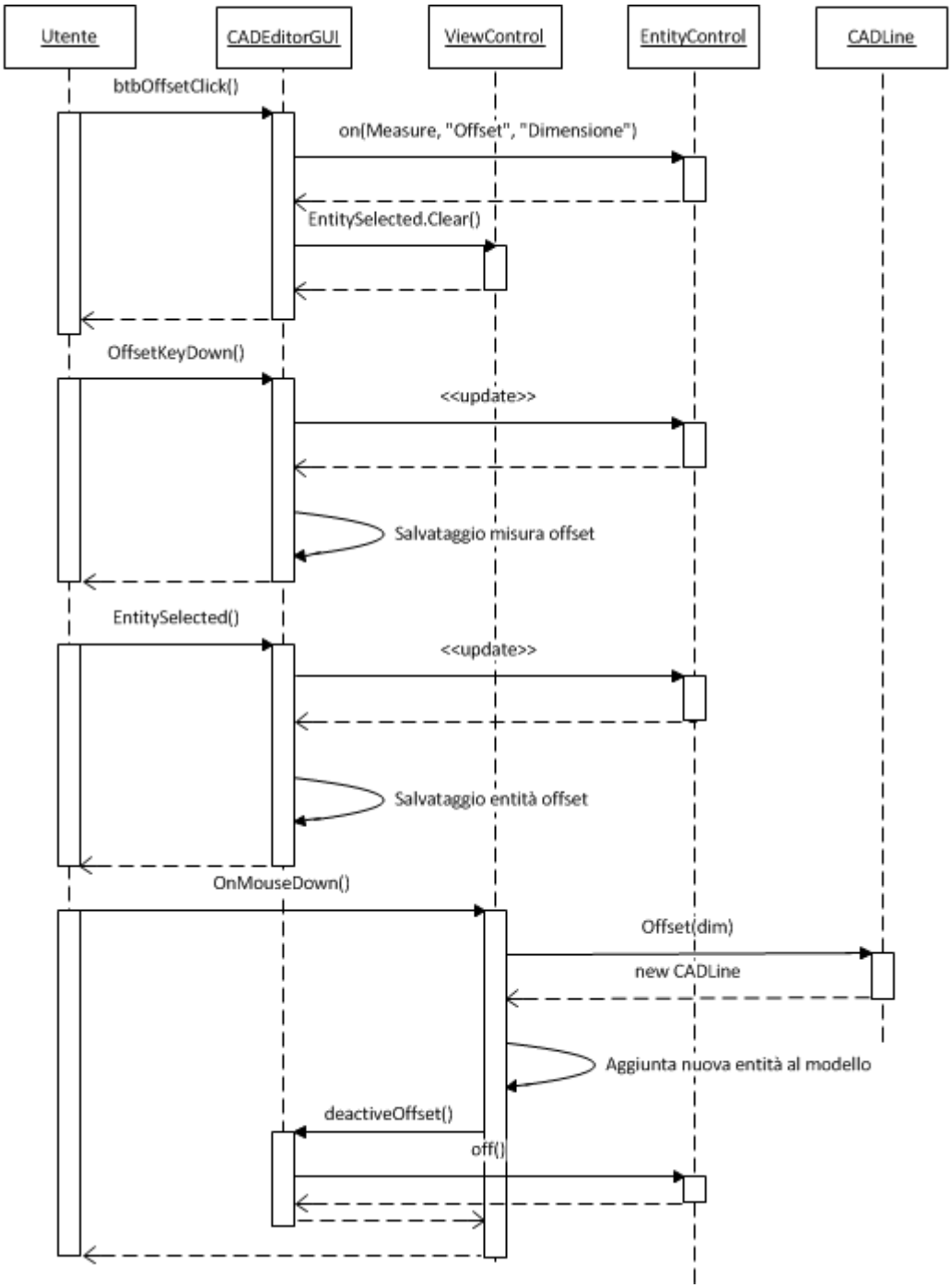

Figura 4.14: Diagramma di sequenza: offset linea.

Capitolo 5

Test

In questo capitolo vengono presentate le varie fasi di test eseguite durante lo sviluppo dell'applicazione. Purtroppo non è stato possibile automatizzare questa procedura e quindi è stata effettuata direttamente da diversi tester indipendentemente, i quali risultati sono stati poi incrociati per la valutazione complessiva dell'unità di test.

5.1 Test delle componenti

5.1.1 Test dell'Interfaccia

In un applicazione come Cad Editor, l'interfaccia ha un ruolo centrale, poiché l'utente interagisce con essa continuamente e deve essere strutturata in maniera tale che le operazioni siano di facile utilizzo e che il numero di click per attivarle sia minimo. Durante lo stage le modifiche apportate sull'interfaccia non sono state molte, sono limitate all'aggiunta di alcuni pulsanti e alla modifica estetica di altri. Come primo step di testing della GUI, si è provato ad attivare tutte le funzionalità, cercando di mettere in evidenza la difficoltà nell'individuare le diverse icone. Da questa fase di test è stato riscontrato che sono necessari al più 2 click per attivare una qualsiasi funzione, questo grazie all'utilizzo della ribbon bar.

5.1.2 Test Creazione

Il test sulla creazione delle entità è stato suddiviso principalmente in 3 sessioni: nella prima le entità vengono create con l'utilizzo del solo mouse[1], nella seconda solo utilizzando la tastiera, nella terza mouse e tastiera vengono usati assieme. Si è notato che l'utilizzo del solo mouse velocizza le varie operazioni a discapito della precisione, mentre con la tastiera la precisione è assoluta. Generalmente le prestazioni migliori si hanno utilizzando entrambi gli input, la tastiera agevola molto la

[1] L'entità Testo deve essere creata inserendo obbligatoriamente il testo da tastiera.

creazione di entità sconnesse tra loro, mentre il mouse diventa molto utile quando le entità presentano punti (marker) in comune.

5.1.3 Test Modifica

Questo test è molto simile al precedente, anche perché la fase di modifica rappresenta in un certo senso uno step intermedio di creazione. Bisogna distinguere però, le modifiche effettuate sulle entità tramite i marker e quelle tramite funzionalità di manipolazione. Nel primo caso i risultati coincidono con quelli dei test di creazione. Per quanto riguarda le altre funzioni, sono state utilizzate su tutte le entità[2] e anche su un insieme di esse e i risultati sono stati quelli aspettati.

5.1.4 Test dei Livelli

Per poter testare il funzionamento dei livelli è stato creato un modello che contenesse almeno un occorrenza di tutte le entità. Poi sono stati creati diversi livelli aventi diverse proprietà (colore, nome, visibilità), i quali sono stati assegnati casualmente alle varie entità. Infine, si è provato a modificare i livelli, aggiungerne di nuovi e a cancellare quelli esistenti, valutando se l'effetto sul modello fosse quello aspettato.

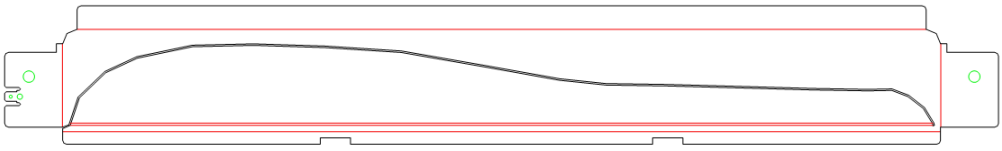

Figura 5.1: Esempio di modello con livelli.

5.1.5 Test dei File

Durante i test precedenti si sono salvati alcuni modelli di prova utili ad effettuare il test sull'apertura dei file. Sono stati inoltre creati modelli con AutoCad che poi sono stati aperti con Cad Editor. Particolare attenzione è stata rivolta alle polilinee, questo perché all'interno del software sono gestite in maniera particolare. Nei

[2] Alcune funzionalità non sono compatibili con specifiche entità. Ad esempio non si può eseguire l'offset di una quotatura.

modelli DXF standard le polilinee hanno un tipo (DxfPolyline2D) diverso da quello utilizzato in Cad Editor (DxfLwPolyline), quindi viene eseguita una conversione.

5.1.6 Test delle altre funzioni

Non sono molte le funzioni che questa fase di test deve controllare. Le più importanti e utilizzate sono sicuramente le funzioni Undo/Redo per annullare o ripetere un operazione. Poi altre funzioni testate sono le misurazioni e la selezione multipla.

Non sono stati eseguiti test sull'esportazione e sulla stampa perché erano funzioni già implementate.

5.2 Collaudo

Per il collaudo finale è stato disegnato un modello basato su un pezzo metallico che viene realmente modellato da robot manipolatori. Questa prova era stata effettuata anche all'inizio dello stage, questo è stato utile a vedere i benefici dello sviluppo dell'applicazione. Una volta creato il modello, in un tempo molto minore rispetto al primo test, è stato salvato e aperto sia con Cad Editor che con AutoCad.

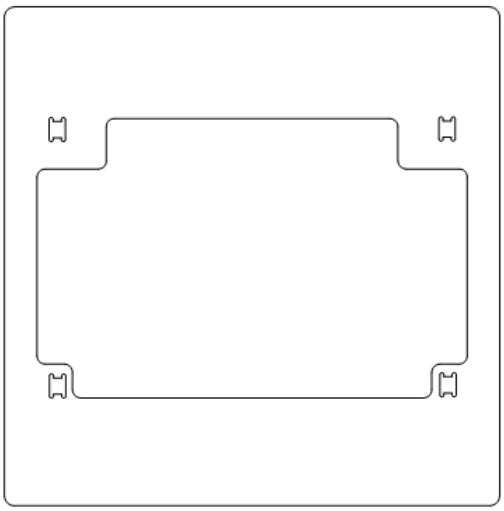

Figura 5.2: Modello di una lamiera.

Capitolo 6
Conclusioni

Le attività svolte all'interno dello stage si sono in parte discostate da quelle definite nel Piano di Lavoro, tanto da poter denominare la presente tesi *Progettazione e sviluppo di un ambiente Cad*, e non semplicemente integrazione di un ambiente Cad. Questo è dovuto soprattutto alla grande libertà che il tutor aziendale mi ha concesso durante lo studio e lo sviluppo di Cad Editor, ed è proprio da questa libertà di inventare che il ruolo dello stage da marginale è divenuto fondamentale nella realizzazione del software. L'applicazione infatti, al di là di ogni aspettativa, viene già utilizzata come plugin all'interno di diverse componenti del progetto SimulEasy v2.
L'azienda ha deciso di creare il pacchetto di installazione del prodotto, in modo da poterlo lanciare sul mercato anche indipendentemente dal progetto in cui è inserito. Questa scelta è stata per me motivo di grande orgoglio e sottolinea ancora una volta quanto l'azienda IT+Robotics prenda seriamente il lavoro dei giovani stagisti (stagiaire).

Per quanto riguarda le tecnologie utilizzate, mi ritengo molto soddisfatto dei tool che l'azienda ha reso disponibile per la realizzazione dell'intero progetto. L'ambiente Visual Studio si è dimostrato ottimo sia nell'integrazione con l'SVN, sia per il debug. L'IntelliSense integrato velocizza notevolmente la stesura del codice evitando i tipici errori di battitura, mentre l'identazione automatica rende il tutto più leggibile e bello da vedere.
Anche l'acquisto della libreria CadLib ha dato grandi benefici nello sviluppo dell'applicazione. Mi ha liberato da molti problemi inerenti la gestione dell'area di lavoro Cad, che altrimenti avrebbero occupato una buona parte delle circa 300 ore a mia disposizione.

Il lavoro di stage mi ha dato la possibilità di vedere come un progetto di grandi dimensioni, qual'è SimulEasy v2, viene gestito, e quanto ogni singola risorsa sia indispensabile per la riuscita di un progetto tanto ambizioso. Ho avuto inoltre la

possibilità di avvicinarmi a tecnologie relativamente nuove, come il linguaggio C#, anche se purtroppo non ho potuto esaminarlo esaustivamente.

6.1 Possibili sviluppi futuri

Nonostante mi ritenga soddisfatto di come è stato realizzato Cad Editor, molte delle funzionalità utili che avrei voluto aggiungere al software non si sono concretizzate. Queste mancanze, segnalate in un apposito documento di fine stage, potrebbero essere prese come spunto per sviluppi futuri. Una su tutte la gestione avanzata dei marker. Nella situazione odierna se più marker si trovano sullo stesso punto, si sovrappongono, e non è definito in alcun modo quale entità venga selezionata cliccando su di essi. Esiste quindi l'eventualità di non riuscire a modificare l'entità desiderata. É auspicabile che in futuro l'azienda, o un nuovo stagiaire, implementi una gestione dei marker più mirata, dando la possibilità all'utente di scegliere quale entità intende modificare. Un'idea da me proposta è quella di visualizzare una lista contestuale, se necessario, contenente tutte le entità che hanno un marker coincidente in quel punto, dando all'utente la possibilità di selezionare quello che desidera.
Anche le funzionalità di taglio e estensione possono essere migliorate, permettendo all'utente di tagliare o estendere più linee con una singola operazione. Lo stesso principio si potrebbe applicare all'offset, permettendo di creare più linee parallele senza dover ogni volta ripetere la procedura di offset. Infine, come riscontrato durante l'analisi di AutoCad, potrebbero essere aggiunte altre modalità di creazione entità, come ad esempio le spline e i poligoni. Le spline soprattutto rappresentano un entità molto importante poiché i modelli possono presentare delle linee curve che, con il Cad Editor attuale, non sarebbe possibile ricreare.

Ringraziamenti

Un grande ringraziamento va al mio relatore, il prof. Paolo Baldan, che nonostante i numerosi impegni mi ha seguito passo passo nella stesura di questa tesi dandomi preziosi consigli. Un grazie anche al mio tutor aziendale, Alberto Conz, e a tutti i componenti di IT+Robotics che mi hanno appoggiato in tutte le decisioni senza crearmi alcun tipo di pressione. Un enorme grazie alla mia famiglia che mi ha sempre sostenuto e incoraggiato in tutto il mio percorso universitario. Infine, ringrazio

tutti i miei amici e Alessandra con i quali ho condiviso i momenti più divertenti e belli di questi ultimi 3 anni.

Appendice A

Glossario

I termini scritti in *corsivo* all'interno della tesi sono chiariti in questo glossario.

.NET: (pronunciato *dotnet*) è un progetto all'interno del quale Microsoft ha creato una piattaforma di sviluppo software, .NET, la quale è una versatile tecnologia di programmazione ad oggetti.

CAD: Computer-Aided Drafting, cioè disegno tecnico assistito dall'elaboratore. I sistemi di questo tipo hanno come obiettivo la creazione di un modello, tipicamente 2D, del disegno tecnico che descrive il manufatto.

DXF: (Drawing Interchange Format, o Drawing Exchange Format) è un formato per i file di tipo CAD, sviluppato da Autodesk come soluzione per scambiare dati tra il programma AutoCAD e altri programmi.

ECMA: (European Computer Manufacturers Association) è un'associazione fondata nel 1961 e dedicata alla standardizzazione nel settore informatico e dei sistemi di comunicazione. Dal 1994 viene chiamata ECMA International.

Estrusione: con il termine estrusione si intende la trasformazione di un modello 2D a un modello 3D equivalente, basandosi su parametri indicati all'interno del modello bidimensionale.

GDI: Graphics Device Interface, anche chiamate Graphics Display Interface (GDI), sono un set di API grafiche utilizzato da Windows per la renderizzazione di grafica 2D.

ISO: (International Organization for Standardization) è la più importante organizzazione a livello mondiale per la definizione di norme tecniche. Fondata il 23 febbraio 1947, ha il suo quartier generale a Ginevra in Svizzera.

Rendering: termine inglese che in senso ampio indica la resa grafica, ovvero un'operazione compiuta da un disegnatore per produrre una rappresentazione di qualità di un oggetto o di una architettura.