# ARCHITETTURA ROBOT THEATRE

# Il Robot:

Triaskarone.

## Sensori necessari:

* Distanza (per il mapping e per l’obstacle avodidance).
* Microfono(facoltativo)
* Sensori di posizione (ancora distanza) da chiedere come si orienta

## Attuatori:

* Ruote e motori: il robot è fornito di movimento holonomico
* Speaker
* Schermo per visualizzare il volto del robot(facoltativo)

## Feature Necessarie:

Mapping: il robot deve poter effettuare il mapping dell’ambiente(il palco)

Localizzazione: costruita la mappa deve sapersi orientare nelle coordinate dello spazio.

Obstacle Avoidance: il robot deve poter indentificare un ostacolo imprevisto e evitarlo

# Spiegazione dei componenti

## Mapping

Il robot deve poter mappare l’ambiente prima dello spettacolo e mostrare sul software (vedi successivo) la mappa del palcoscenico.

Potrebbe dover mappare anche più scene dato che potrebbero cambiare.

Probabilmente serve un algoritmo di SLAM, ma è da chiedere come fare e se c’è un altro modo per avere la mappa della scena.

Idea: usare I landmark e dire al robot a che distanza li posizioniamo. Perché i landmark? Imporrebbero una spesa ulteriore per posizionarli. Vero, ma una scena deve essere provata dal regista più volte e spesso si ha accesso al teatro solo un giorno prima della recita, quindi ci sarebbe poco tempo per fare il mapping e verificare le posizioni del robot, per questo è meglio usare i landmark che simulano le dimensioni del palco anche se non si è in teatro.

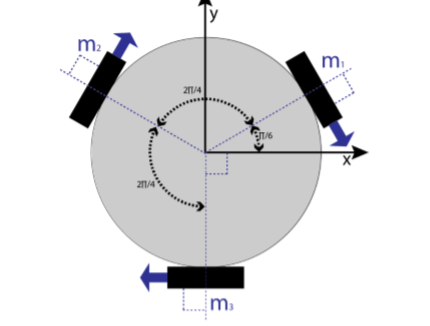
Chiaro che il posizionamento dei landmark non sarà mai preciso al mm, ma questa cosa si può risolvere con delle fuzzy rules, in modo tale che la posizione possa essere valida se è più o meno quella (pochi centimetri non cambiano)

I landmark devono essere posizionati anche sugli oggetti di scena fissi, non è un grosso problema dato che di solito non sono tantissimi. Però il professore suggerisce di orientarsi con il mapping, bisogna provare.

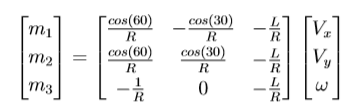
## Movimento

### MOTOR CONTROLLER

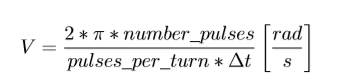
Il robot è costituito da 3 ruote olonomiche che gli permettono di muoversi in tutte le direzioni.



Dette Vx Vy e w le velocità desiderate, le velocità da fornire ai motori sono



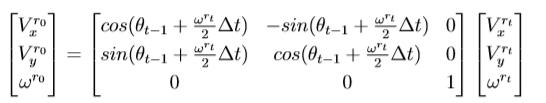
Dove R=MISURARE e L=MISURARE (R raggio della ruota e L distanza dall’asse)

Il nodo in ros avrà quindi in ingresso la velocità desiderata e darà potenza ai motori in modo tale da muoverli a quella velocità, in particolare V (velocità motore):

In cui bisognerà scegliere il AT (in base a a prove) e “pulses per turn” è il numero di pulsazioni per fare un giro (dato da ricavare

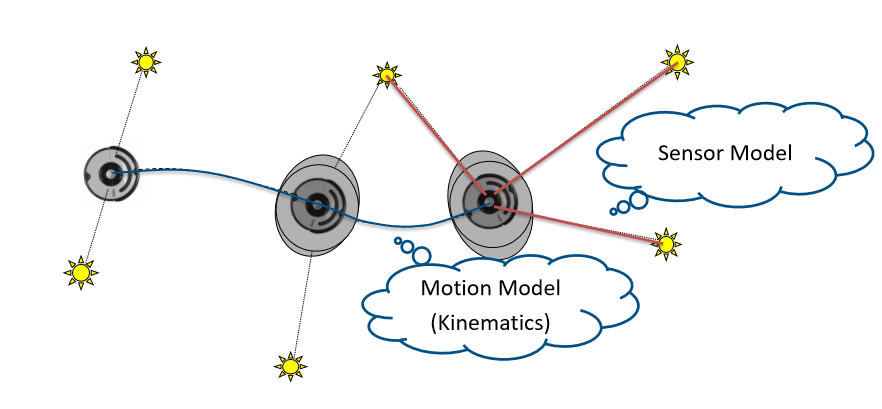
### NODO CONTROLLO VELOCITà (FACOLTATIVO)

Siccome la velocità effettiva non è mai quella reale (a causa di attriti e slittamenti) il nodo avrà in ingresso la velocità desiderata e le misurazioni fatte e pubblicherà quella effettiva



NON HO CAPITO MOLTO è DA CHIEDERE, in teoria si potrebbe usare solo il nodo successivo

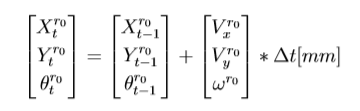
### POSITION CONTROLLER



Questo nodo calcola la posizione attuale unendo i dati della cinematica con i dati dei sensori di posizione.

La mappa è nota, ma non è sempre precisa al 100%, bisogna gestire questa cosa (vedi obstacle avoidance).

#### CALCOLO POSIZIONE (Kinematics):



Data la posizione precedente, la velocità e il tempo trascorso si calcola la posizione in cui si trova

#### CALCOLO POSIZIONE( Sensori)

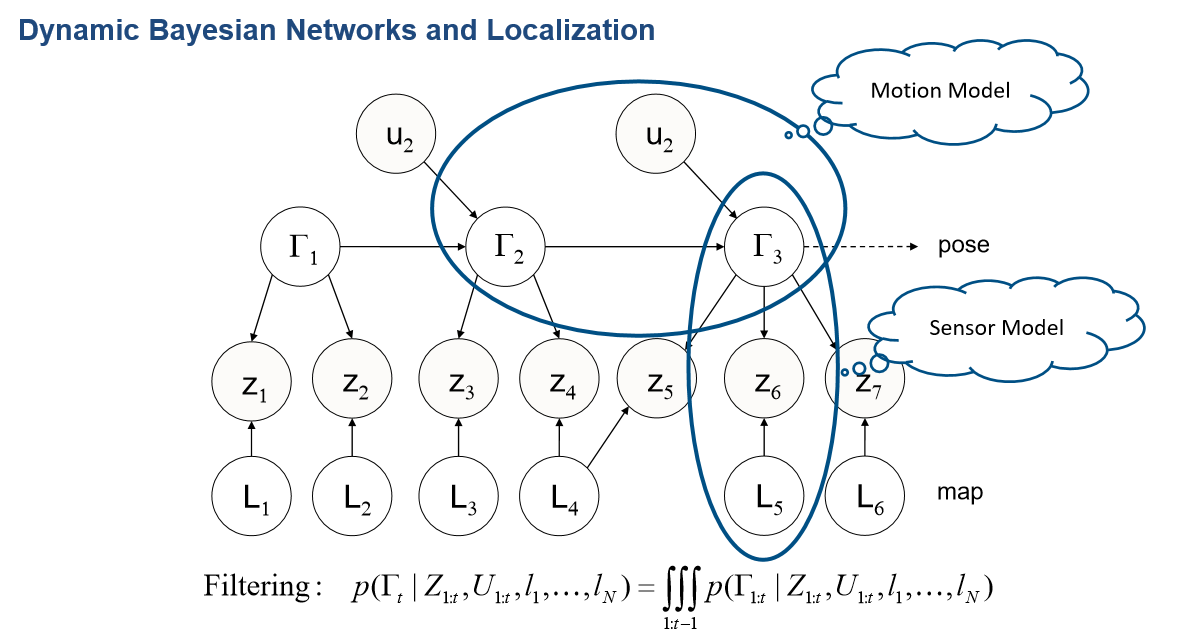
Dai sensori di posizione legge la posizione

Possibili algoritmi:

* Scan matching (non ha bisogno di altro, ma meno preciso)
* Landmark(più preciso, ma abbiamo bisogno di avere dei landmark)

#### CALCOLO POSIZIONE (FINALE)

Combinando dati di cinematica e posizione va a trovarsi la posizione nello spazio e la pubblica sul topic: **“/actual\_pose”**



Possibili Algoritmi:

* Bayesian Filtering
* Kalman Filter
* Particle Filter

Entrambi si basano su predirre la posizione in cui saremo e aggiornarla in base ai dati dei sensori.

### OBSTACLE AVOIDANCE

In teoria lo spazio dovrebbe essere libero, ma sul palcoscenico potrebbero capitare i seguenti problemi:

* Un attore potrebbe sbagliare posizione e mettersi sulla traiettoria del robot (si potrebbe risolvere semplicemente fermandosi in modo che l’attore possa capire di aver sbagliato)
* Un oggetto di scena potrebbe essere involontariamente nella posizione sbagliata

Per quest’ultima opzione soprattutto bisogna permettere al robot di evitare l’ostacolo e sembrare il meno possibile impacciato, per questo una buona opzione è “Dinamic Window Approach” perché ci permette di operare anche in spazi stretti, cosa abbastanza comune in scena, ma non sarà questo nodo a implementare l’algoritmo (vedi dopo)

### PATH PLANNING

La parte più complessa. Il regista deve poter scegliere da che punto a che punto della scena il robot deve muoversi o muoversi verso uno specifico attore.

Il nodo quindi programmerà tutti i punti intermedi che il robot deve raggiungere indicando la velocità per ogni tratto. Il moto dovrà essere scelto in base alle emozioni provate dal robot in quel momento, l’algoritmo verrà mostrato in seguito

La traiettoria può essere divisa in sottotratti perché l robot potrebbe anche muoversi avanti e indietro lungo di essa, mostrando un atteggiamento insicuro.

La traiettoria dovrebbe essere possibilmente curva e deve essere eseguita in maniera più naturale possibile, per questo avrei bisogno di aiuto su questa parte sui calcoli da fare

### PATH EXECUTION CONTROL

Questo nodo controlla l’esecuzione del tratto di percorso da eseguire, infatti fatti i dovuti calcoli su velocità e tempi potrebbero esserci imprevisti come slittamenti o ostacoli, questo nodo si prende cura di ricevere il sottotratto da eseguire dal path planner e ha cura di eseguirlo nei tempi stabiliti, aumentando o diminuendo la velocità se necessario o fermandosi (desired\_velocity=0) nel caso si riceva un segnale di ostacolo dal nodo obstacle avoidance ricalcolando la traiettoria.

## Sensori

### Tracciamento attori

Il robot deve poter conoscere la posizione degli attori sulla scena, per questo le soluzioni proposte sono due:

* Riconoscere gli attori con una telecamera (magari in base al colore del vestito), soluzione non invasiva, ma più complessa
* Ogni attore porta con se un piccolo microcontrollore che trasmette al robot posizione e nome dell’attore (soluzione più semplice da implementare e quella consigliata)

Per ora suppongo che l’attore rimanga comunque fermo in un punto e non si muove

### Tracciamento Discorso

Il robot deve poter parlare per recitare le proprie battute, ma deve anche dirle al momento giusto seguendo il copione. Per poterlo fare esistono due soluzioni:

* Riconoscere il discorso dell’attore, questa soluzione è la più “autonoma”, ma presenta alcune problematiche
  + Il riconoscimento del linguaggio non è sempre preciso
  + L’attore umano potrebbe sbagliare la battuta mandando il robot in confusione
* Parlare quando gli viene dato un comando esterno, le criticità sono le seguenti:
  + Perde in parte il senso di automazione
  + Serve una persona che indichi al robot di parlare, può essere l’attore con un radio bottone o il regista allo stesso modo.

### Bottone di emergenza

Il robot deve poter bloccare tutti i suoi nodi in caso venga premuto il bottone di emergenza (sul robot e sul joystick) per semplificare tutti i nodi si iscrivono a questo topic e in caso venga premuta bloccano tutto e si mettono in stato di attesa. L’esecuzione dei task deve poi poter riprendere dal punto in cui era (possibilimente).

## Attuatori

### Speaker

Il robot deve poter parlare seguendo lo script, ci sono due modi per farlo:

* Registrare la voce di un vero attore e mandare i file audio
* Implementare un sintetizzatore vocale

Nel caso della seconda scelta serve un nodo che genera le tracce audio in base all’emozione del robot, per adesso non progetto questa parte.

### Schermo

Il robot non avrà un volto umano e tantomeno muscolatura del volto, dato che sarebbe troppo complicato e costoso implementarlo. Per dare espressività però si potrebbe installare in cima al robot uno schermo che mostri diversi volti stilizzati (in stile robotico) e anche dare un’impressione del parlare.



## Braccia

Si potrebbero aggiungere delle semplici braccia che fanno un movimento rotatorio, aspetto per metterlo nello schema.

## Busto

Il robot dovrebbe poter muovere il busto avanti e indietro per aumentare ulteriormente la capacità espressiva

## NODO CENTRALE

Si è seguita una politica di divisione dei compiti in modo da decentralizzare il più possibile le sottofunzioni, ma c’è bisogno di un nodo centrale che coordini i vari movimenti del robot.

Il nodo centrale quindi legge lo script e comunica ai vari nodi incaricati il compito da svolgere, si preoccupa di ricevere l’avvenuta esecuzione delle azioni e gestisce gli errori.

# Software

Un regista non è un programmatore quindi bisogna scrivere un software che permette al regista di programmare la scena.

## Fase 1: Definizione spazi scena

Il software chiede se si vuole iniziare una nuova scena.

Chiede la posizione dei landmark e mostra a video la mappa del palco con questi landmark

## Fase 2: Movimenti

Il regista ora con un semplice click del mouse unisce i vari punti dove si deve muovere il robot.

Per ogni coppia di punti il robot chiede se deve pronunciare una frase durante il movimento e chiede l’emozione attuale del robot, il regista avrà una barra degli strumenti

Il robot non potrà mai uscire dalla zona segnata a parte un permesso speciale per uscire di scena in punti precisi della mappa.

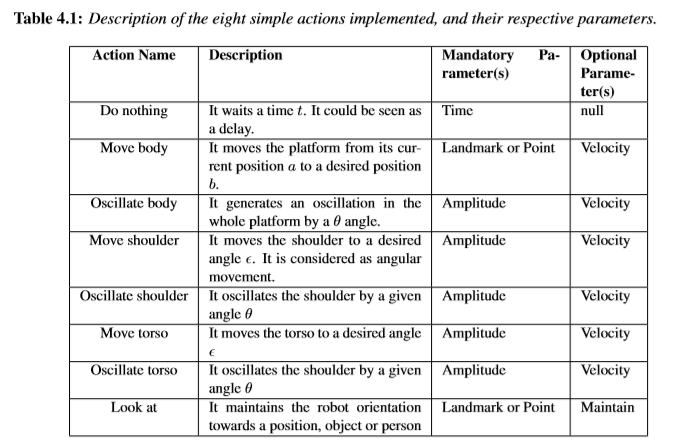
A questo punto può fare una simulazione del movimento (questo non saprei come programmarlo)

Salvata la scena può caricare la scena sul robot.

# Azioni del robot

Le azioni possono essere definite sia semplici che complesse come descritto nella tesi di Fernandez, il registra potrà selezionarle come vuole, sarà computo del software gestirle e suddividerle in varie sottoazioni da mandare all’unità di controllo, che a sua volta le gestirà contemporaneamente.

Qui c’è un esempio di azioni semplici, ma verranno modificate a seconda delle disponibiiltà del robot.



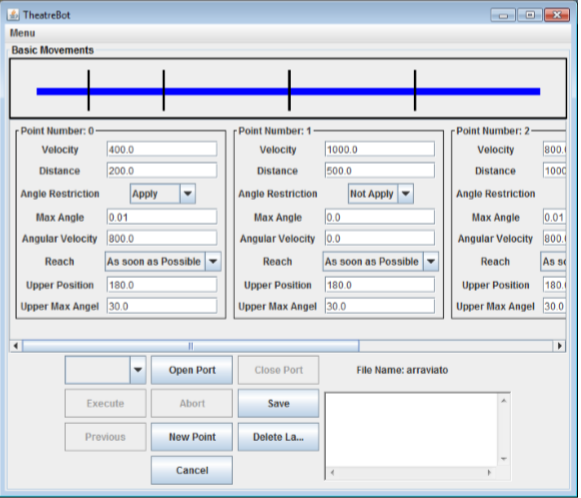
# 

# Algoritmi

Bisogna scegliere ora come rappresentare le emozioni, che è al centro del nostro miglioramento di quello che, finora, è un robot capace “solo” di muoversi nello spazio.

## Opzione 1

Dare al regista tutte le responsabilità facendo scegliere movimenti e velocità punto per punto



## Opzione 2

Le emozioni per ogni instante saranno definite dal regista, è anche possibile creare una struttura di interpretazione in futuro, ma per ora proviamo a implementare una struttura più semplice

Si divide in due parti: modulazione e generazione:

* Modulazione: si definiscono i movimenti da fare e l’emozione al momento
* Generazione: il software genera i movimenti arricchiti in base ai parametri forniti e li scrive in file xml che sarà il nostro script robotico.

NON HO CAPITO BENE LA PARTE CHE DICE CHE SELEZIONA UN’AZIONE DI RISERVA, perché? NON MI BASTA ESEGUIRE L’AZIONE CORRETTA?

L’architettura di Fernandez è distribuita, mentre io pensavo di farne una centralizzata cosa cambia? Oppure si può fare una struttura centrale a sua volta definita da più azioni.

### Definizione azioni

Azioni complessi possono essere composte da azioni semplici usando formule logiche, ogni azione ha dei parametri da definire, facciamo un esempio:

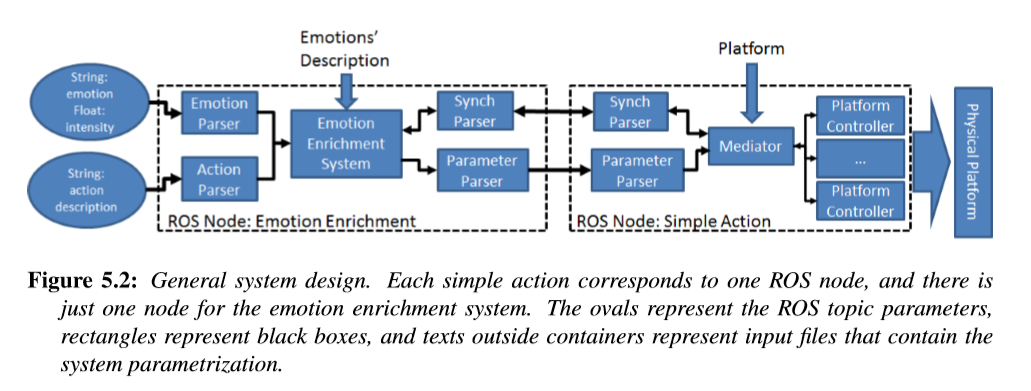
azione semplice: *Param(move\_body) = {position,v,ω}*

azione complessa:

* *Param(walk) = {position,v,angle}and Parallel(balance\_arm(left,angle,ω1) balance\_arm(right,−angle,ω1) move\_body(position,v,ω2))*
* *Param(move\_talk) ={position1, position2,angle1,angle2,text} move\_talk =Parallel(Sequence( walk(position1,v,angle1), walk(position2,v,angle2)), speak(text,pitch)*

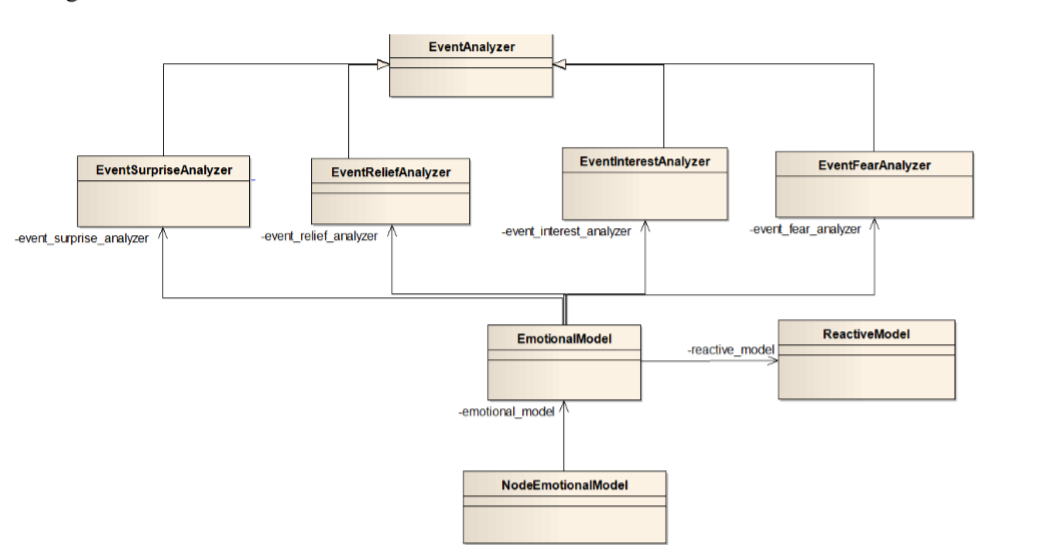
Parallel indica che le azioni vanno eseguite in parallelo, sequence che vanno eseguite in sequenza

### Arricchimento



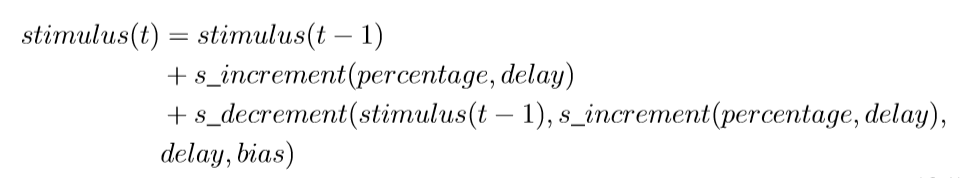
Non ho capito benissimo come funziona, cioè io lo progetterei con un nodo che chiama più nodi che arricchiscono ogni singola azione e poi questo nodo manda tutto a esecuzione. Ancora si parla di azioni fondamentali e altre no e non capisco il senso.

### Selezione delle emozioni

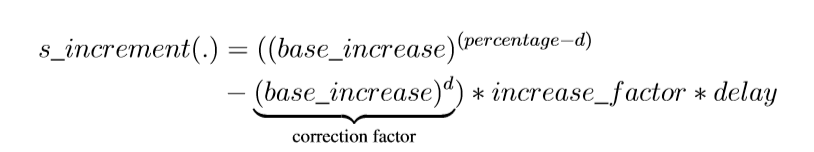


Il modello di Fernandez parla di un analizzatore di emozioni, cioè più “nodi” analizzano un aspetto dell’emozione e causano un incremento o un decremento di questa emozione, è una soluzione possibile, ma un’altra soluzione potrebbe essere far selezionare aumento e decremento dell’emozione da parte del regista.

Comunque l’incremento è fatto così:



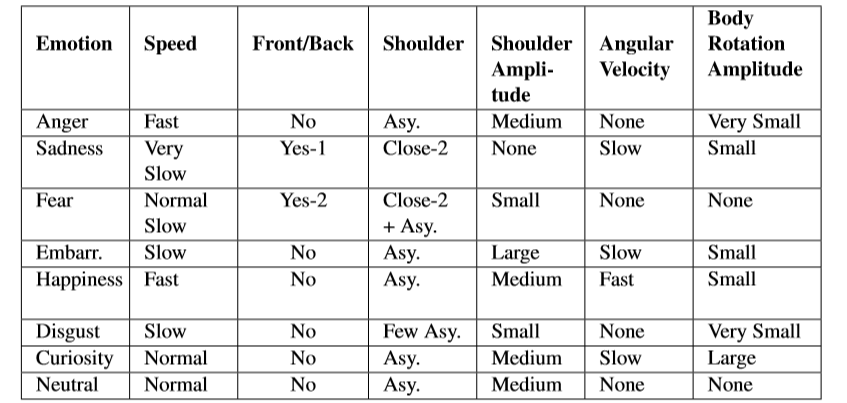
Dove



Tutto questo produce un emozione che deve andare ad arricchire l’azione, in che modo?

### Correlazione con il movimento

Si può definire una tabella e in base ad alcune fuzzy rules produrre il movimento finale



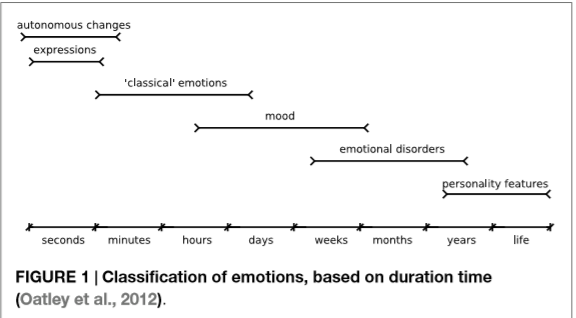
Un’altra opzione però potrebbe essere rendere modificabile questa tabella, che può rappresentare il carattere del robot, ad esempio un robot può avere “sangue freddo” e quindi essere veloce quando prova paura o rimanere totalmente bloccato, si deve poter quindi modificare questa tabella.

# Selezione emozioni

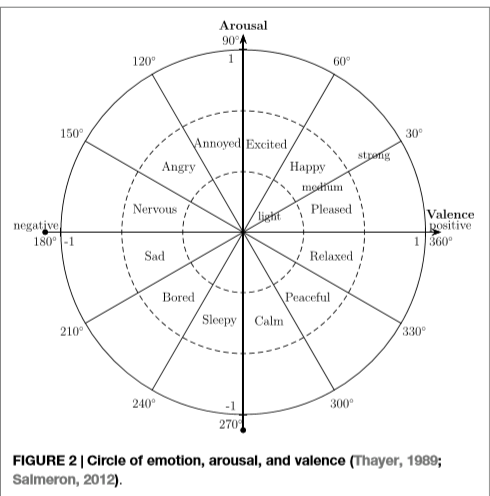
Il robot deve avere memoria delle emozioni precedenti in modo tale da non avere un cambio repentino, ma provare emozioni in base a tutte le emozioni provate prima con un forgetting factor.

Idea potrebbe essere dare una selezione di emozioni disponibili per ogni scena e fare assegnare un valore, questo va a cambiare il valore precedente e di conseguenza l’emozione provata.

Altra idea sarebbe dare un carattere al robot, in modo che provi emozioni in relazioni al suo carattere (vedi pagina 3 documento ESE:

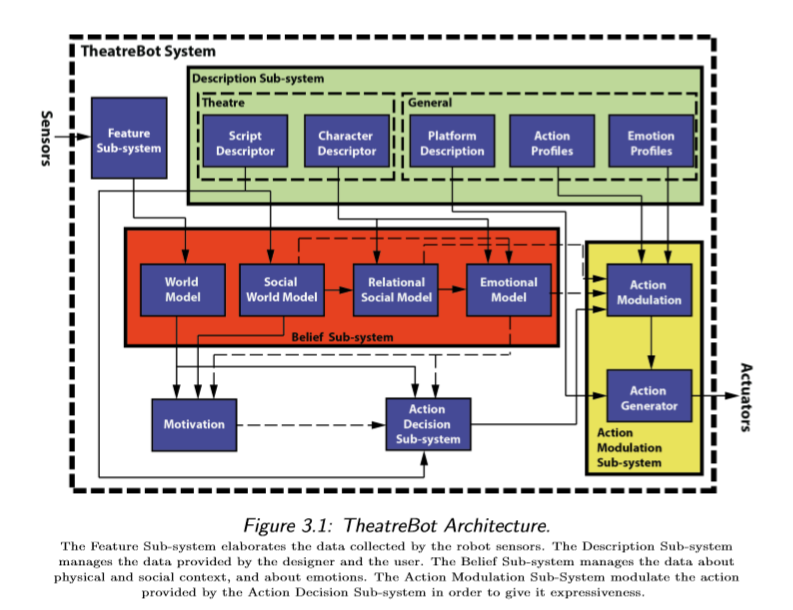


Si utilizzeranno per modellare le Fuzzy rules, l’architettura FLAME usa emozioni opposte, tipo gioia e tristezza.



Le fuzzy rules potrebbero essere imparate, ma per rendere la cosa più semplice le hardcoderei

# Architettura TheatreRobot

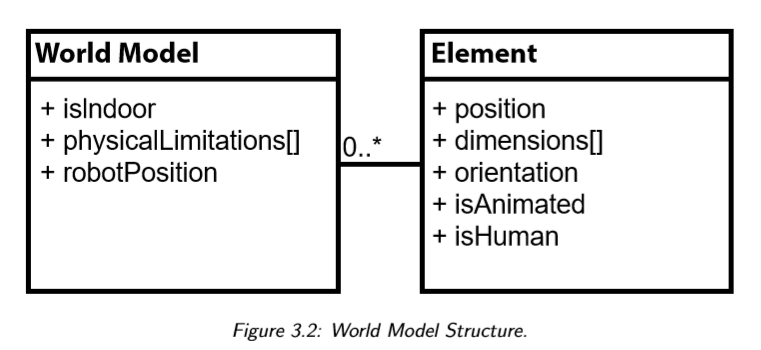


1. **Feature Subsystem** basandosi sui dati dei sensori crea una rappresentazione del mondo in cui è
2. **Describtion Subsytem** descrive il mondo attraverso i dati forniti dai designers e i programmatori
3. **Belief Subsytem** ha una descrizione del mondo fisico e del contesto sociale e ha il modello delle emozioni
4. **Motivation** ha le necessità e i desideri degli attori
5. **Action decision** contiene le decisioni da prendere basandomi sullo script
6. **Action Modulation** dà espressvità

## World Model

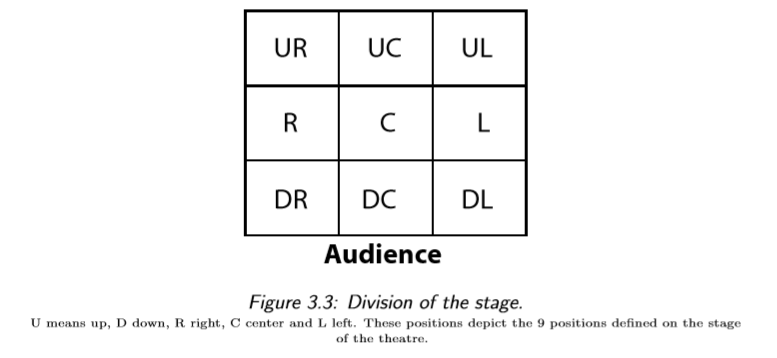
Contiene informazioni fisiche dell’envviromnet, in particolare:

* Mappa della scena
* Posizione sulla mappa (ottenuta con i sensori)
* Posizione degli attori sulla scena
* Se il robot è umano o no



## Script

Contiene le **azioni che deve fare il robot**, in particolare il luogo in cui si deve muovere è diviso in nove aree



Inoltre contiene la **descrizione dei personaggi**

## Social World Model

Contiene le informazioni sui sentimenti tra i diversi personagg e i loro gruppi , è da considerarsi come l’elemento di base per partire con lo spettacolo, poi man mano con gli eventi relazioni e sentimenti cambiano

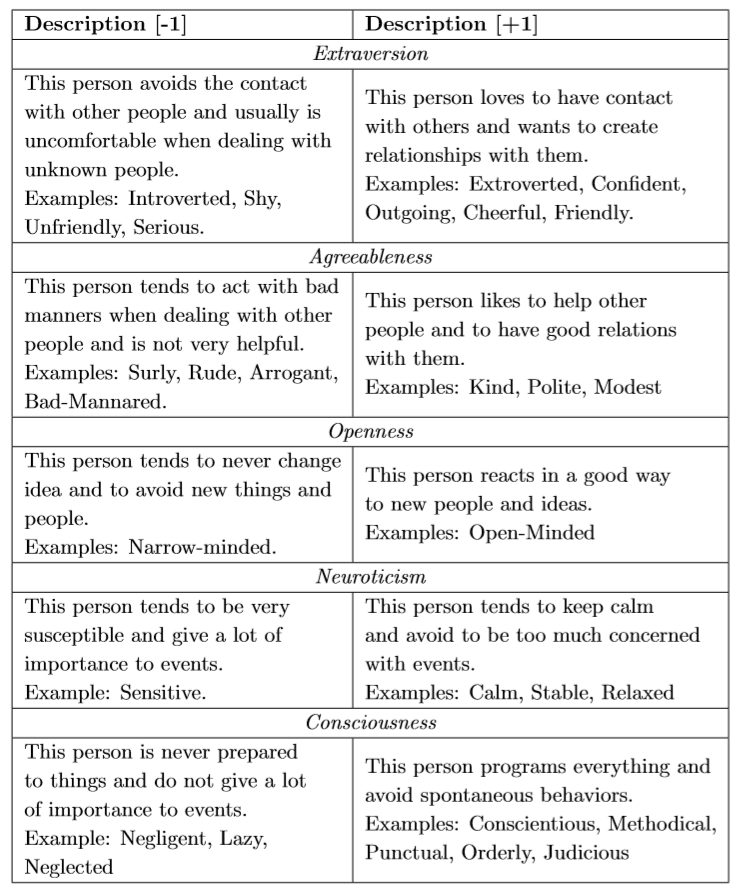
## Social Entity

Descrive il singolo personaggio , in paricolare:

* **Disposition**: indica quanto il personaggio può cambiare
* **Personal Trait**: indica il carattere del personaggio e in che modo reagisce agli eventi
* **Mood**: come i personal trait, ma è più dinamico in base al momento

I personal traits sono distinti in questi 5 fattori, a cui si assegna un valore da -1 a 1

1. **Extraversion**, denoting the way in which the person interacts with the external world. An extrovert person tends to seek the company of others, while an introverted person is deﬁnitely more reserved and reﬂective
2. **Agreeableness**, denoting in a general way if the person is well-tempered or not. Low agreeableness leads to a more suspicious personality, while high agreeableness leads to a more kind personality
3. **Openness**, denoting the degree of curiosity and appreciation for new experience and ideas. An open-minded person tends to react in a good way to new ideas, indeed he looks for them, while a narrow-minded person tends to reject any new experience;
4. **Neuroticism**, denoting the degree of emotional stability. A neurotic person tends to be extremely susceptible and to experience easily emotions like anxiety or depression. Moreover, his mood tends to change very quickly. On the contrary, a stable person tends to be calm and relaxed, and his mood does not change that much;
5. **Conscientiousness**, denoting the capability to self-discipline. A conscientious person programs everything and prefers planned activities rather than oﬀhanded ones. A careless person is never prepared to events and prefers to behave spontaneously in most of situations.



Allora la **DISPOSITION** viene calcolata in questo modo:



Dove:

* P è la personalità
* M è il mood
* Ws è la forza del mood, cioè quanto influenza la disposizione, nel valore da 0 a 1
* Wd è il decay factori del mood

Un altro fattore importante è la **Conditionabilty,** vista in seguito

## Relazioni

Sono le connessioni tra i vari personaggi

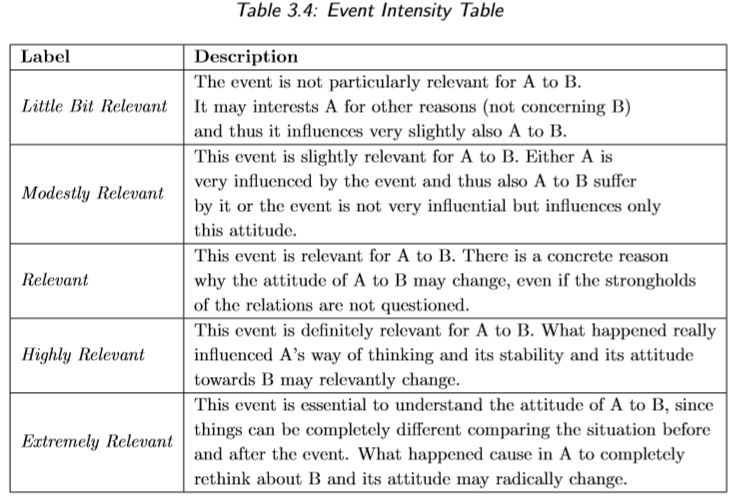
Una relazione può essere:

1. **Symmetric** Relations. In this case the relation is perceived at the same way in both directions. This means that, just from the Social World Model point of view, the sentiments of the two entities for the other one are the same;
2. **Antisymmetric** Relations. In this case the relation as perceived from one entity to the other is exactly the opposite if seen from the other direction. This means that the sentiments of the two entities for the other one are the opposite;
3. **One-Way** Relationships. In this case the relation as perceived from one entity and the same relation as perceived from the other entity have no relationship. This can mean that the two sentiments are not related, or even that the relation subsists only in one of the two direction (it may be that the second entity does not even know the ﬁrst one, or at least he ignores him).

a ogni relazione è associato un **sentimento,** che è a lungo termine (a differenza dell’emozione) , vedi tabella pagina 34.

## Eventi

Possono cambiare mood e carattere se conosciuti dal persoaggio, gli eventi possono influenzare in questo modo:

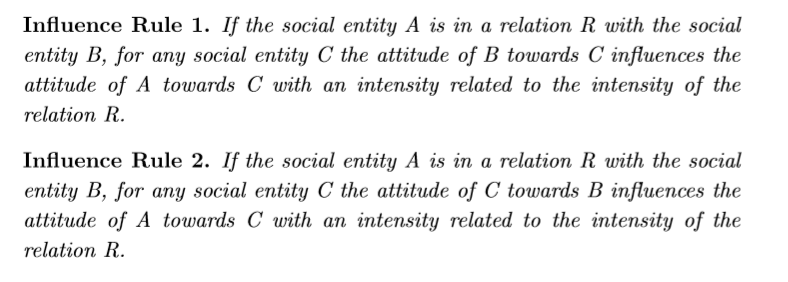


## Reletional Social Model

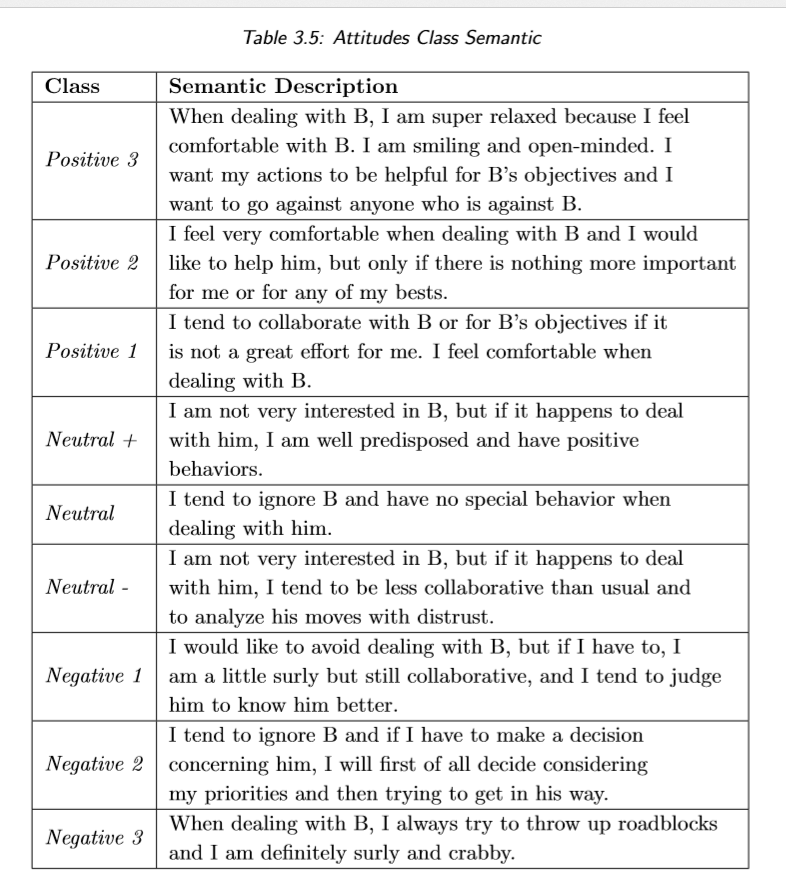
Fa due cose:

1. Descrive come i le attitudini che provano le persone verso gli altri influenzano gli altri
2. Come gli eventi influenzano le attitudini cioè il comportamento che un personaggio ha verso un altro

1: si usa questa regola finchè non si raggiunge un equilibrio



2: è più complicato perché rompono l’equilibrio



# Struttura JSON

Per rappresentare lo script è stato scelto un file json, esso deve dividersi per ogni scena e ogni scena deve dividersi in ogni tratto, potrebbe essere strutturato in questo modo:

* #Scena
  + Tratto1
    - Movimento?
      * Verso attore?
      * Posizione sulla scena
      * Nome attore
      * Velocità
    - Busto
      * Avanti e indietro?
      * Velocità
    - Voce
      * Battuta?
      * Frase da dire
      * emozione
  + Tratto2

# Domande per il professore

* Posso indicare solo la velocità? O serve anche l’accelerazione?
* Ho ulteriormente pensato al sintetizzatore vocale facendo lezione di teatro oggi, secondo me non è una buona soluzione perché nonostante si possa dare un emozione fissa rimane il problema dell’interpretazione teatrale parola per parola, che non è piatta come un normale discorso. Soprattutto per i monologhi, ma comunque può essere lasciata come opzione.

# Appunti sui possibili movimenti delle emozioni

1. **Curiosità:** il movimento è lento verso l’oggetto, il busto in avanti, gli occhi sbattono velocemente, il collo si piega di lato
2. **Rabbia:** Busto in avanti, movimento veloce occhi socchiusi, bocca stretta

## Scelte finali per i file

Due file:

1. Scritp: definisce tutti i movimenti che deve fare il personaggio, dividendo l’opera in atti scene e sezioni, ogni sezione ha movimenti, emozione e quando viene triggerata
2. Character: definisce il personaggio, cioè parti mobili, tabella dei movimenti e pesi alle regole

L’idea è quindi scrivere un interprete che legge la sezione e in base a cioè che è scritto nel carattere definisce il movimento finale