

**Autore Ing. Antonio Di Fluri**

**In collaborazione con**



**[www.futuresoftware.it](http://www.futuresoftware.it)**

**[info@futuresoftware.it](mailto:info@futuresoftware.it)**

# Sommario

DESCRIZIONE.....	3
STRUMENTI DA UTILIZZARE.....	3
APPROCCIO AL PROBLEMA.....	4
Per ottenere la massima generalità rispetto alle applicazioni possibili, lo stream di dati avrà il seguente formato: .....	4
1. ANALISI E SPECIFICA DEI REQUISITI.....	6
Figura 1: Use Case Diagram .....	6
2. PROGETTAZIONE.....	7
Figura 2: View of Packages .....	7
Figura 3: Collaboration Diagram .....	8
Figura 4: Sequence Diagram.....	9
3. IMPLEMENTAZIONE.....	10
Figura 5: Classes of service.....	10
Figura 6: Classes of event .....	10
Figura 7: Classes of test .....	11
Figura 8: Class Diagram.....	12
4. TESTING E VALIDAZIONE .....	13
Figura 9: menu principale e menu preferenze xlet.....	13
Figura 10: Tris Game e Simple Server.....	13
Figura 11: Fine del gioco (sconfitta).....	13
Figura 12: Gantt Diagram .....	14

## DESCRIZIONE

---

Il software da sviluppare consiste della parte lato client di un framework completo per la gestione di comunità.

Il framework comprenderà oltre alla parte client anche un lato server che dovrà interagire con i vari client, così da offrire agli utenti servizi di virtual community.

L'utente, e quindi il client, dovrà per prima cosa impostare la configurazione per un determinato servizio, quindi comunicarla al server. Qui la configurazione sarà gestita, cioè inizialmente creata e, successivamente, memorizzata se modificata.

## STRUMENTI DA UTILIZZARE

---

- Librerie Java TV
- Librerie MHP
- Java socket
- Java thread
- Eclipse
- Fujaba
- Xlet View
- OpenMhp (utilizzo della console per il debug)

## APPROCCIO AL PROBLEMA

---

Il primo problema si è rivelato essere l'ideazione di un protocollo di comunicazione tra il client e il server, che fosse generale e non limitante.

Si è poi dovuto studiare il funzionamento del “canale di ritorno” della piattaforma MHP, necessario per poter mandare dati dal client, o meglio dalle applicazioni dei client, al server; in realtà questo canale viene utilizzato anche dal server per mandare dati alle applicazioni client.

Per quanto riguarda questo secondo problema, la soluzione adottata è stata quella di stabilire la connessione del dispositivo STB con l'ISP locale, ottenendo quindi una connessione a Internet, e successivamente di aprire una socket verso il server che gestisce i dati del servizio di Community.

Per quanto riguarda il primo problema invece, una volta collegati al server, la comunicazione sarà basata su uno stream di dati continui, strutturati con un particolare formato che permetterà lo scambio di informazioni e parametri qualsiasi.

Per ottenere la massima generalità rispetto alle applicazioni possibili, lo stream di dati avrà il seguente formato:

*<tipo dati trasmessi>[<carattere delimitatore><nome dato><valore dato>(1..n)]<carattere fine stringa>*

Un esempio di applicazione, coerente con il progetto in esame, potrebbe essere il seguente:

```
preferences#mossa1#23#mossa2#12|
```

preferences: indica che quella che segue è una stringa che definisce le preferenze dell'utente.

#: carattere che delimita le coppia nome-valore cioè nome=mossa1, valore=23

|: carattere che indica la fine della stringa.

I caratteri delimitatore e fine stringa devono essere impostati all'inizio della comunicazione.

L'utente può scegliere a piacimento tutti i caratteri purché questi non siano all'interno del nome e del valore.

In merito a questa ultima limitazione si potranno prevedere politiche di byte stuffing.

Per permettere una più semplice implementazione della classe di comunicazione, anche lato server, sarà d'obbligo stabilire al momento della realizzazione del software i caratteri delimitatori, e non consentire più all'utente di cambiarli. Si cercheranno ovviamente caratteri che consentano la maggiore trasparenza possibile verso l'utente, che si potrà quindi disinteressare completamente di essi.

Nel dettaglio, questo documento è strutturato secondo l'approccio utilizzato durante lo sviluppo:

- Analisi e specifica dei requisiti
- Progettazione
- Implementazione
- Testing e validazione

## 1. ANALISI E SPECIFICA DEI REQUISITI

---

RF1: il sistema deve impostare i parametri per la comunicazione con il server (indirizzo IP ed eventualmente porta TCP di comunicazione, username e password per l'accesso al sistema).

RF2: il sistema deve impostare le preferenze per l'applicazione (xlet)

RF3: il sistema deve inviare le preferenze

RF4: il sistema deve inviare lo stato aggiornato dell'applicazione

RF5: il sistema deve ricevere il nuovo stato per l'applicazione

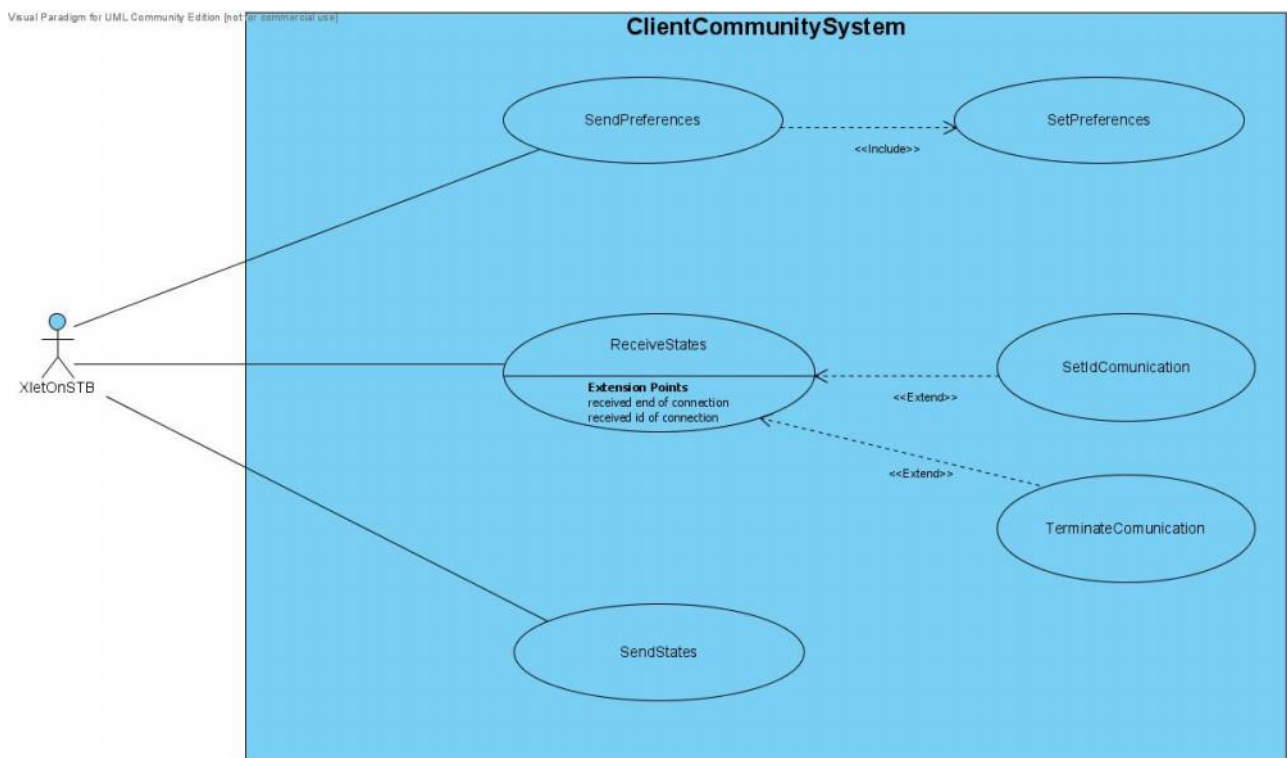


Figura 1: Use Case Diagram

## 2. PROGETTAZIONE

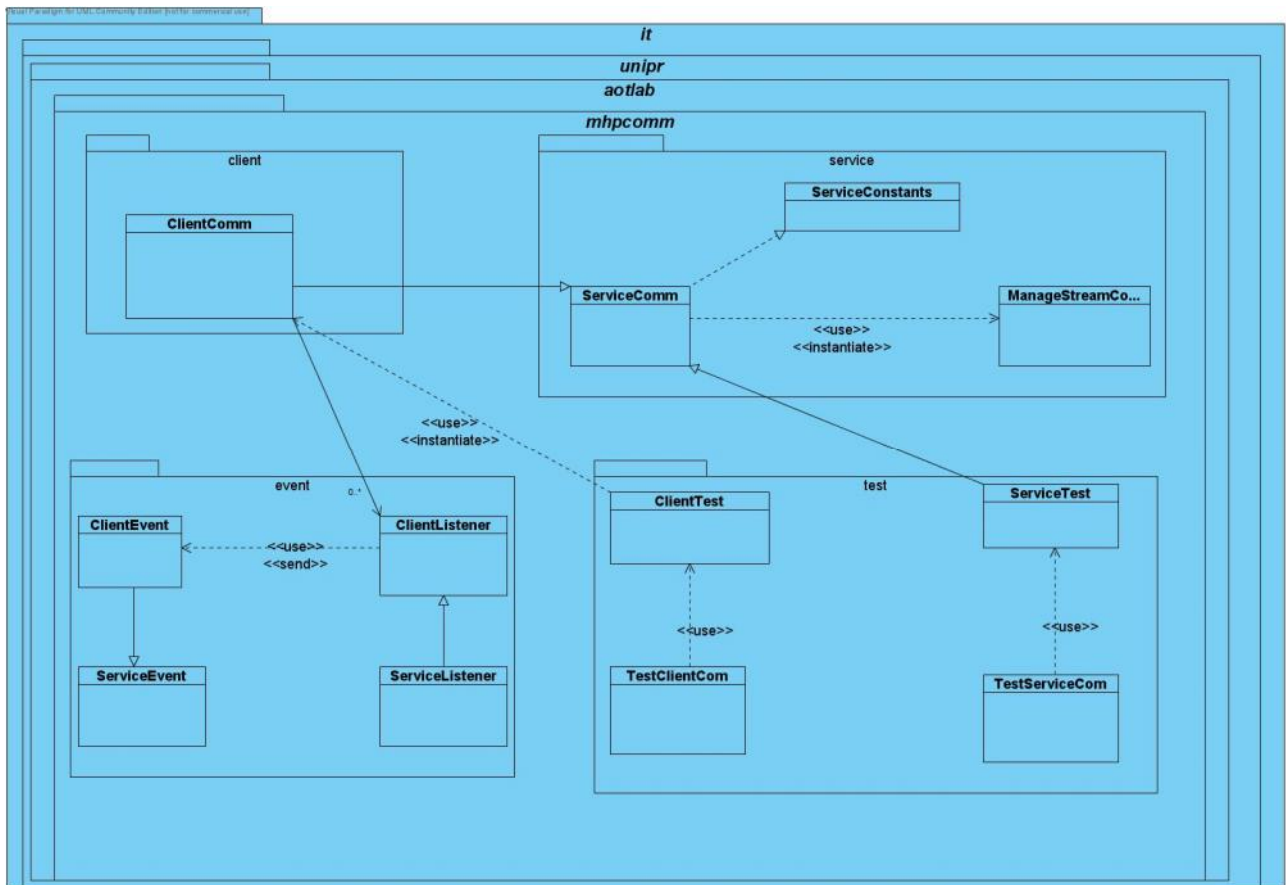


Figura 2: View of Packages

Nella fase progettuale si è data fondamentale importanza alla genericità del sistema da realizzare, così da non limitare la varietà di applicazioni che potranno appoggiarsi ai servizi di community offerti.

Innanzitutto bisogna ricordare che la community sarà strutturata da uno o più server centrali e da una moltitudine di client, non necessariamente su piattaforme MHP, che attraverso il dialogo coi server potranno interagire tra di loro.

Il sistema qui progettato è la parte comune a tutte le applicazioni della community: esso dovrà far “parlare” le applicazioni con il server centrale e dovrà girare sulla piattaforma MHP.

In primo luogo è stata progettata una classe **ClientComm** che, dopo aver presentato all’utente richiesta di connessione, si collega al server e richiede all’utente stesso quale applicazione eseguire.

Una volta istanziata la classe relativa all’applicazione, che dovrà implementare l’interfaccia **ClientEvent** e **ServiceEvent**, quest’ultima prenderà

il controllo dell'interfaccia grafica e potrà contare sui metodi per comunicare col server offerti dalla ClientComm.

Il sistema verrà infine corredato di un package per il test delle singole funzionalità.

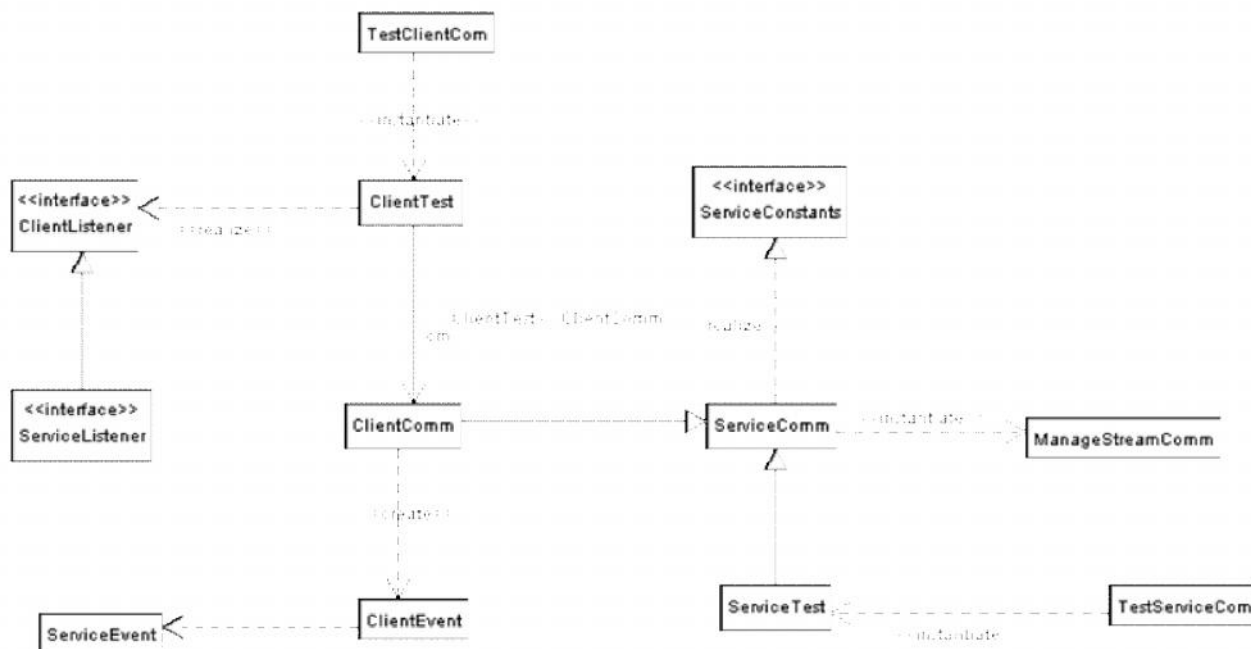
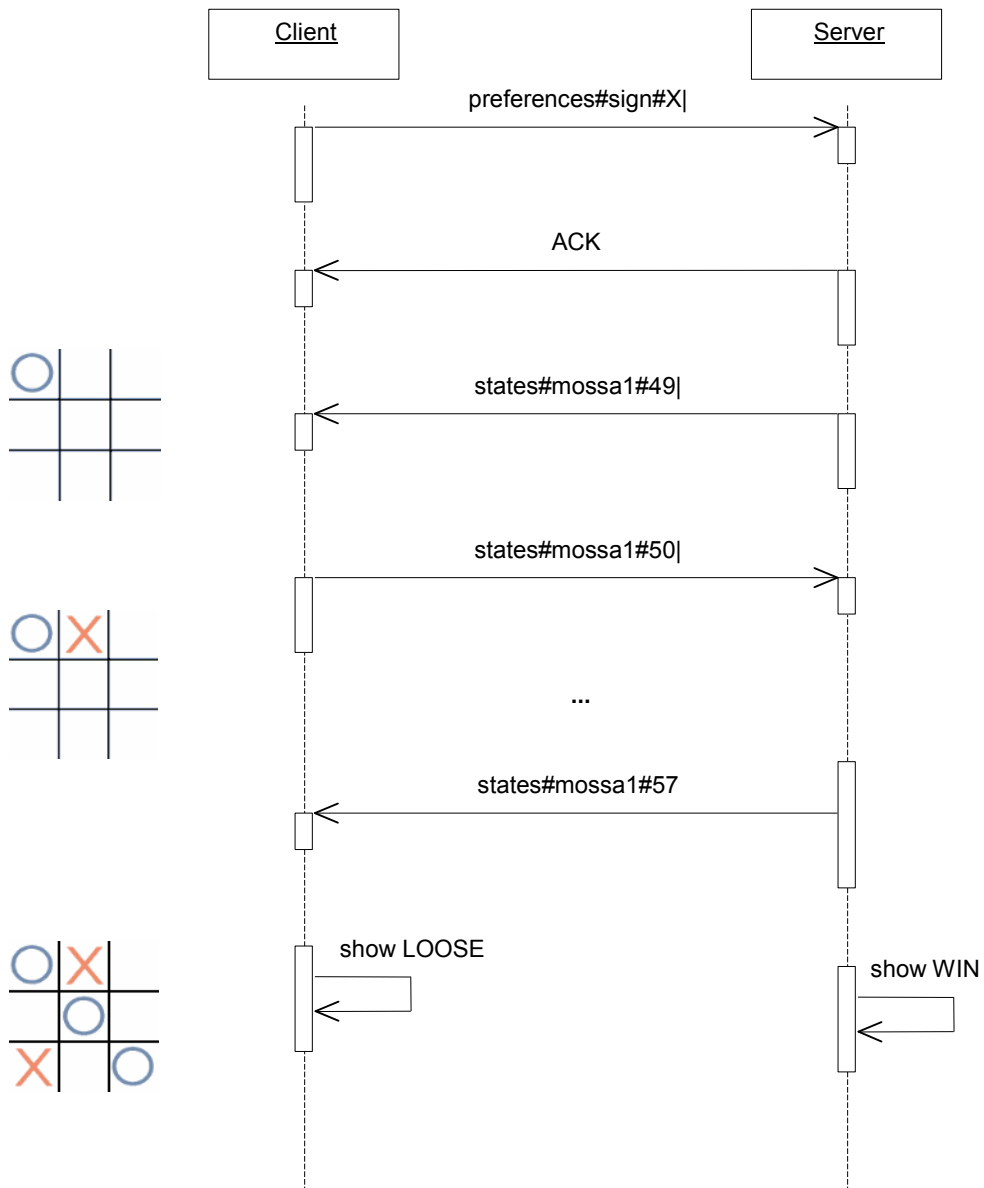


Figura 3: Collaboration Diagram



**Figura 4: Sequence Diagram**

### 3. IMPLEMENTAZIONE

Per ciò che riguarda l'implementazione, si sono incontrate alcune difficoltà nel riuscire a far funzionare il canale di ritorno della piattaforma MHP (poiché anche difficoltoso da simulare con gli emulatori).

Il problema è stato risolto sfruttando la documentazione reperita riguardo alla piattaforma MHP e alle librerie JavaTV.

Per dettagli di implementazione si rimanda al codice sorgente.

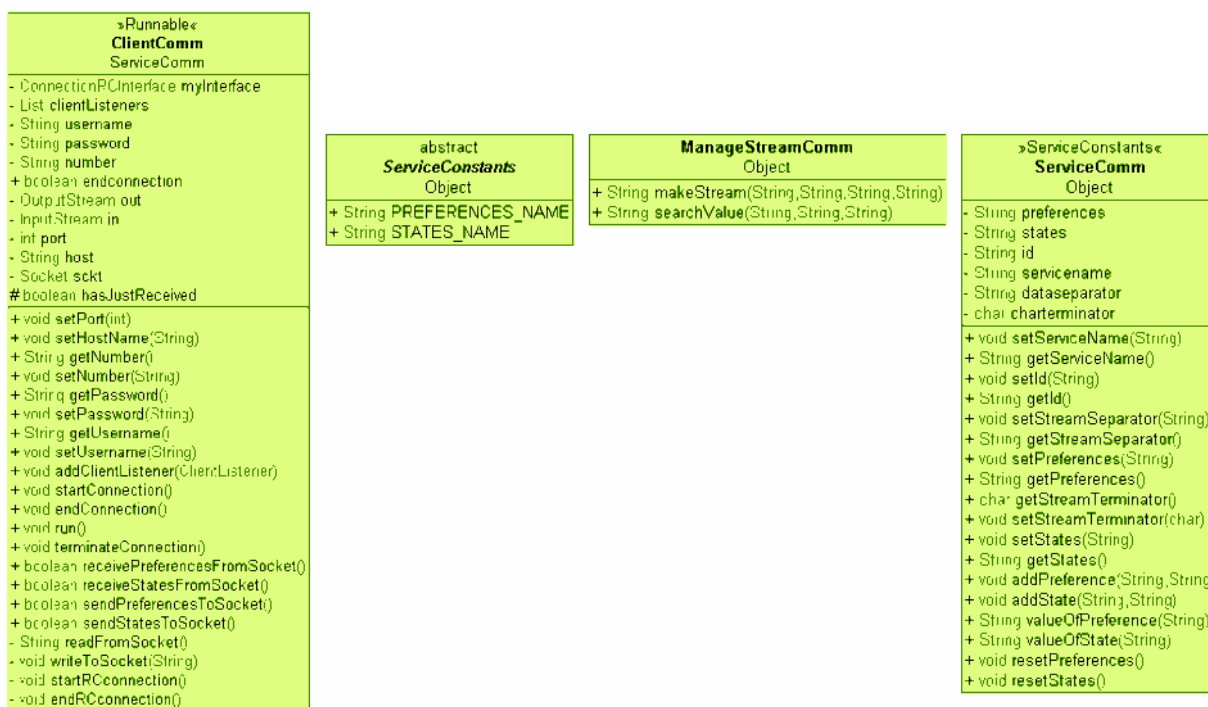


Figura 5: Classes of service

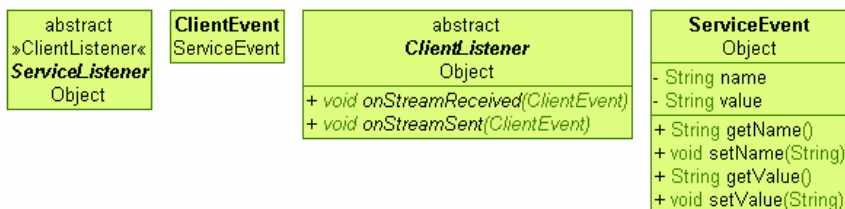
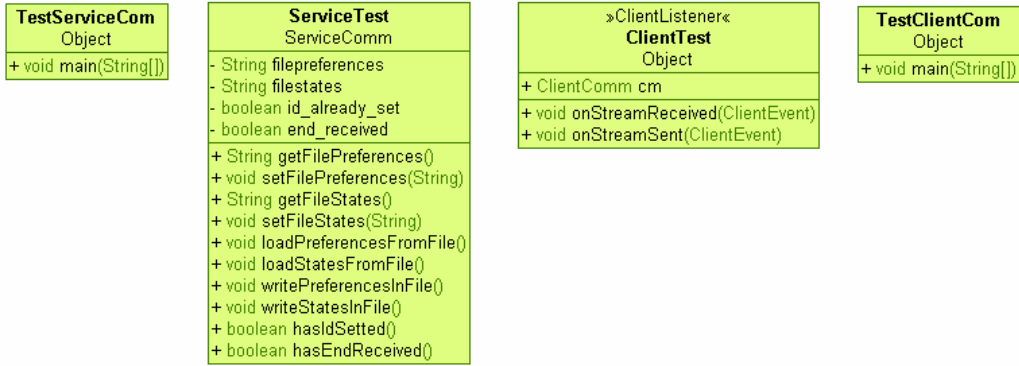


Figura 6: Classes of event



**Figura 7: Classes of test**

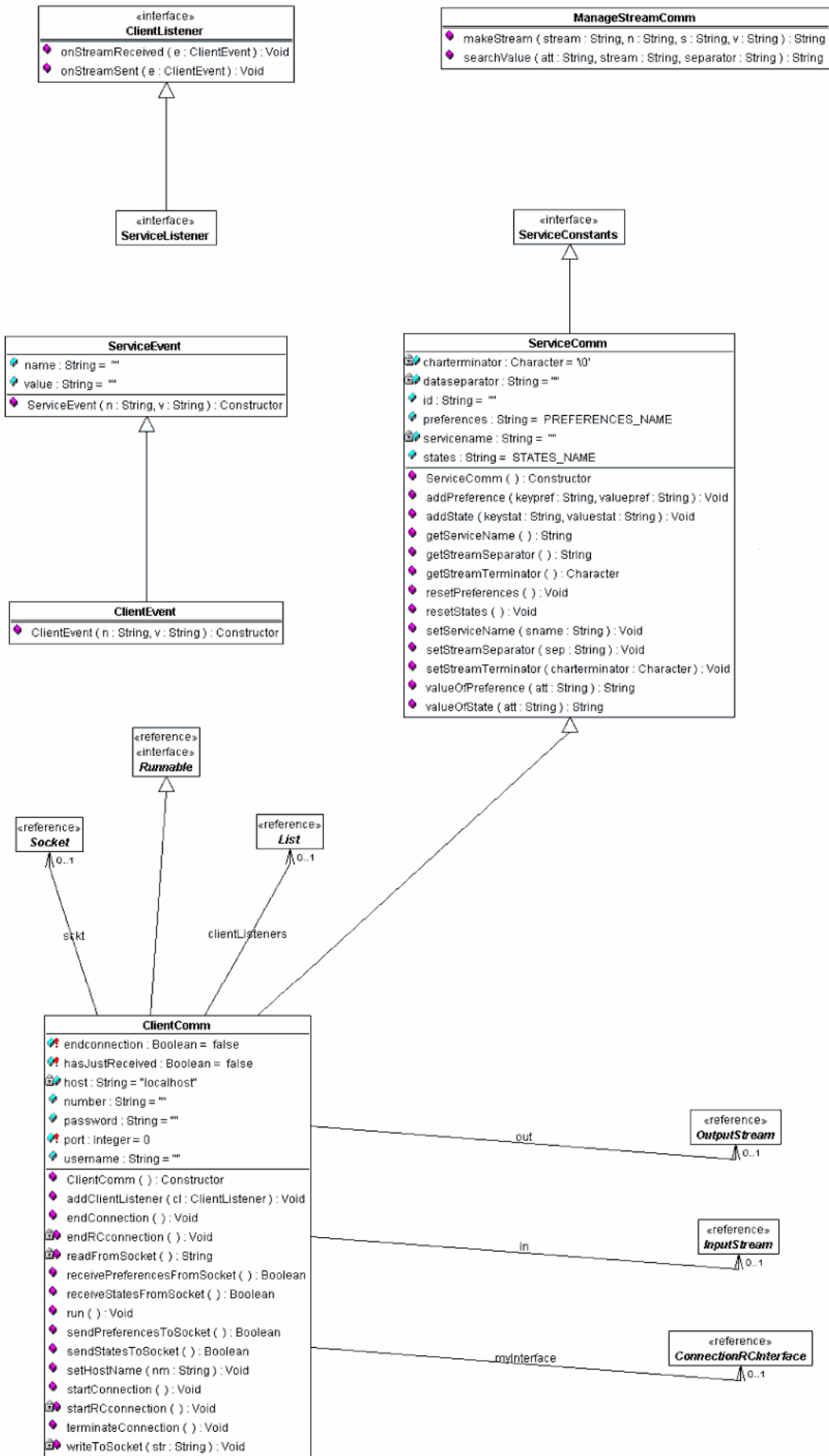


Figura 8: Class Diagram

#### 4. TESTING E VALIDAZIONE

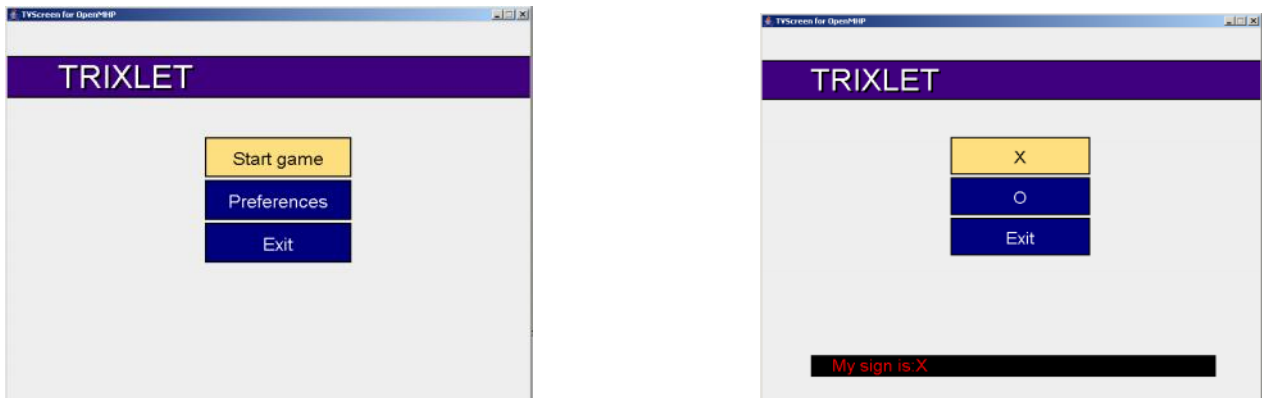


Figura 9: menu principale e menu preferenze xlet

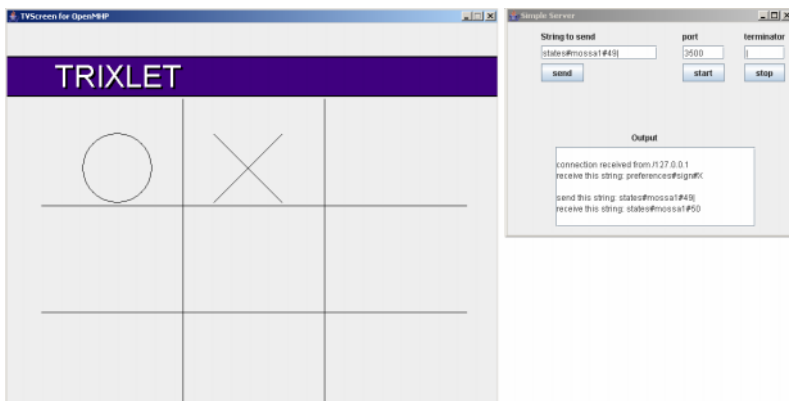


Figura 10: Tris Game e Simple Server

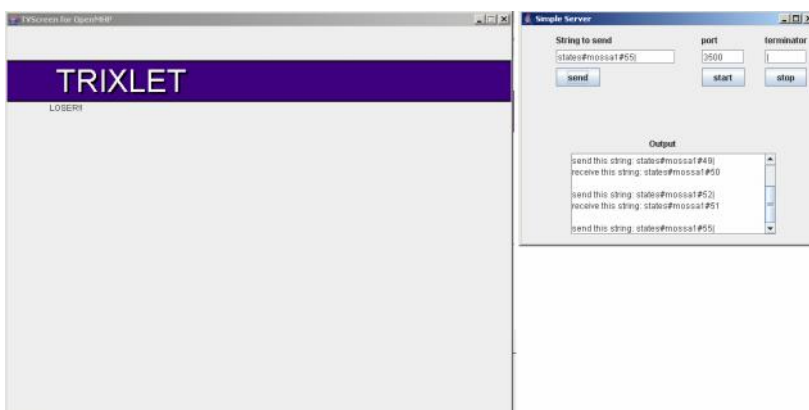


Figura 11: Fine del gioco (sconfitta)

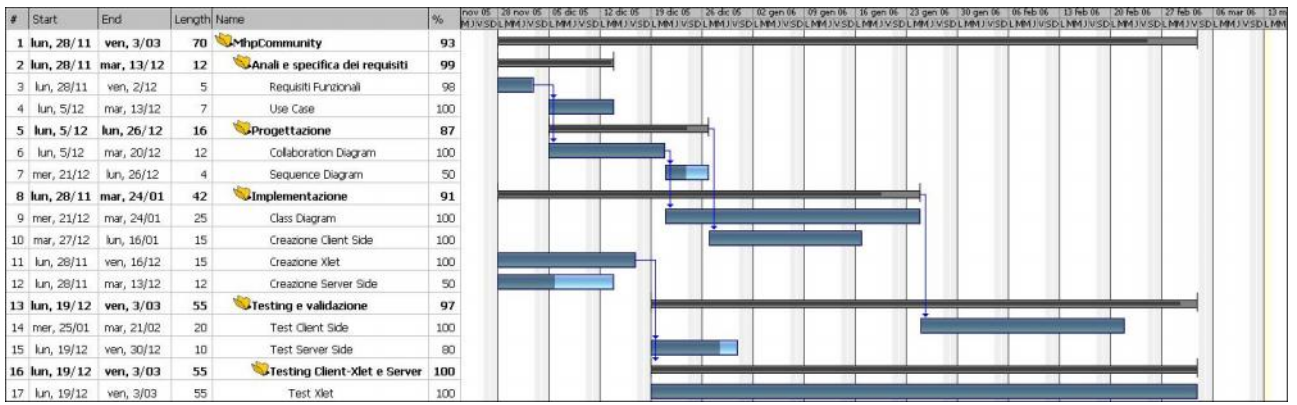


Figura 12: Gantt Diagram