Progetto di sistemi intelligenti distribuiti L-S

2005-06

Recita di Poemi Famosi

Ovvero come implementare un MAS di agenti java attraverso Jack & TuCSoN

Relazione

di

Maggioli Tommaso

Gli obbiettivi

Nella realizzazione di questo progetto si è cercato di perseguire diversi fini. Si è partiti cercando di sviluppare un primo sistema multi-agente distribuito attraverso una piattaforma commerciale, JACK. Poi, dopo aver visto i progetti dei colleghi, si è provato a sviluppare un sistema equivalente attraverso TuCSoN. Così facendo si è giunti a voler confrontare le tecnologie TuCSoN e JACK attraverso le due implementazioni sviluppate

L'idea, il contesto

Dopo aver ammirato in televisione la bravura con cui Roberto Benigni recita a memoria e con passione la Divina Commedia di Dante Alighieri, ho provato a realizzare qualcosa di simile. Così facendo ho cercato di produrre qualcosa di socialmente-culturalmente utile, realizzando una sorta di recita ad agenti che riproponga qualche classico della nostra letteratura.
Una motivazione aggiuntiva che mi ha spinto ad intraprendere tale strada è stata il fatto che tra i progetti passati disponibili on-line non ho visto nulla di simile.

La piattaforma JACK

JACK rappresenta una piattaforma commerciale per lo sviluppo, l'esecuzione e l'integrazione di sistemi multiagente tramite un approccio a "componenti". Esso è un prodotto che scaturisce dal lavoro nel campo della ricerca e dello sviluppo delle tecnologie per gli agenti da parte del Agent Oriented Software Group (AOS). Quest'ultimo rappresenta uno dei più grandi gruppi internazionali per lo sviluppo di software per sistemi orientati agli agenti.
Il linguaggio di programmazione utilizzato in JACK estende quello di Java con concetti propri della teoria degli agenti, come ad esempio quelli di: Agenti, Capacità, Eventi, Piani, Basi di Conoscenza per Agenti, Risorse e Gestione della Concorrenza.
JACK rappresenta il nucleo architettonico e infrastrutturale per lo sviluppo e il lancio di software ad agenti in sistemi distribuiti.

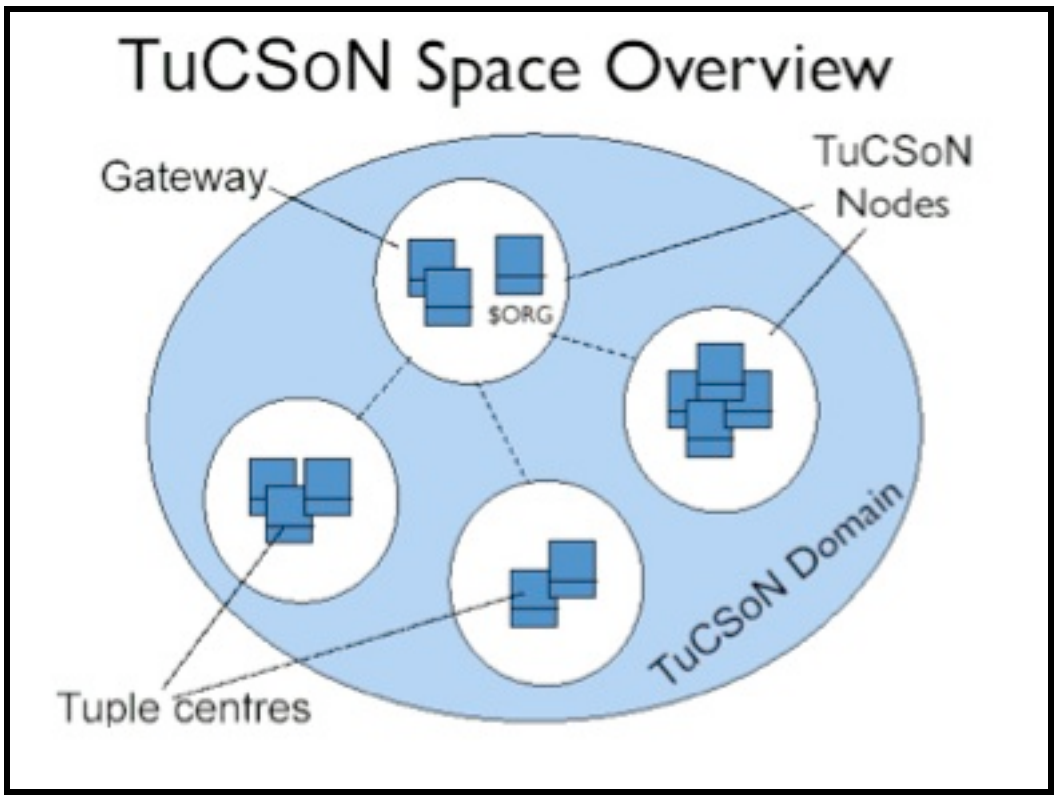
JACK consente un modello di comunicazione leggero e flessibile che permette, via TCP/IP, di far interagire agenti con altri agenti e con GUIs.
AOS ha utilizzato un approccio modulare nello sviluppare JACK. Questo consente all'utente di selezionare i soli componenti della piattaforma, di cui ha bisogno.
JACK, nella sua totalità, contiene i seguenti elementi:

- Un ambiente di sviluppo
- Un ambiente di esecuzione
- Un compilatore
- Modelli per agenti BDI
- Display per l'interazione tra gli agenti
- JACOB: tool per l'interazione con altri processi o applicazioni esistenti

Particolarità di pregio di JACK sono gli ambienti grafici di sviluppo. JACK infatti incorpora una suite di strumenti grafici pensati sia per analisti che per programmatori. Sia chi si trova a dover sviluppare una applicazione sulla base del paradigma ad agenti, sia chi deve analizzare la correttezza di progetti già esistenti, potrà utilizzare degli intuitivi menù grafici per il proprio lavoro.

TuCSoN

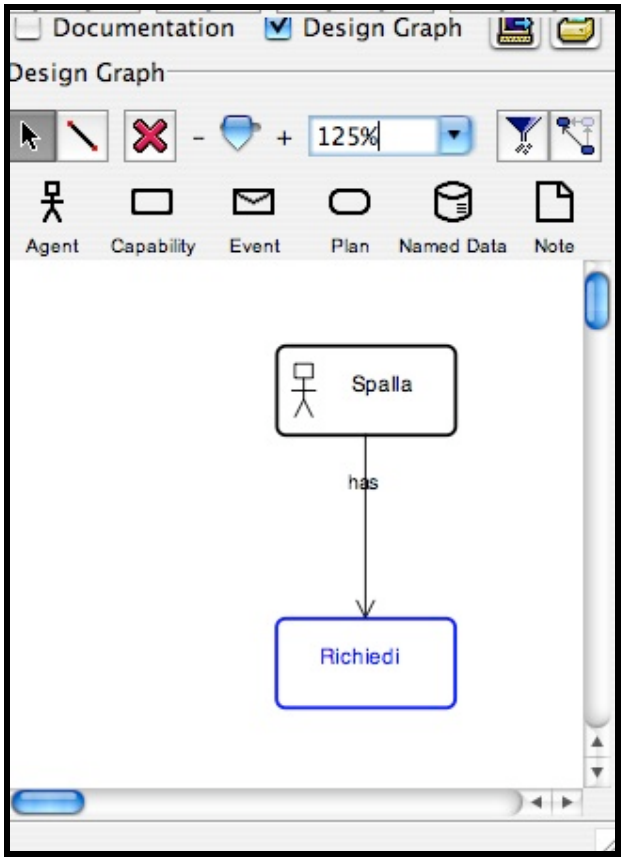
TuCSoN `e un'infrastruttura di coordinazione per MAS. TuCSoN, si basa sull'utilizzo di centri di tuple logiche (tuple centre o TC) come astrazioni di coordinazione di prima classe, identificati da un nome. Tuple centre sono raggruppati in nodi. I nodi, che sono entità IP- indirizzabili, sono catalogati in 'places' e 'gateway'. L'organizzazione del sistema TuCSoN prevede che raggruppamenti di più places e di un gateway costituiscano dei 'domini'.
E' possibile che ciascun nodo faccia parte di più domini, assumendo anche diverse veci (places da una parte, gateway dall'altra). I gateway hanno funzionalità superiori rispetto ai places. Essi contengono, nei loro tuple centre delle tuple logiche riguardanti l'organizzazione del dominio di cui fanno parte. L'insieme dei domini forma il mezzo di coordinazione di TuCSoN. L'organizzazione di tale infrastruttura puo' essere rappresentata come in figura.



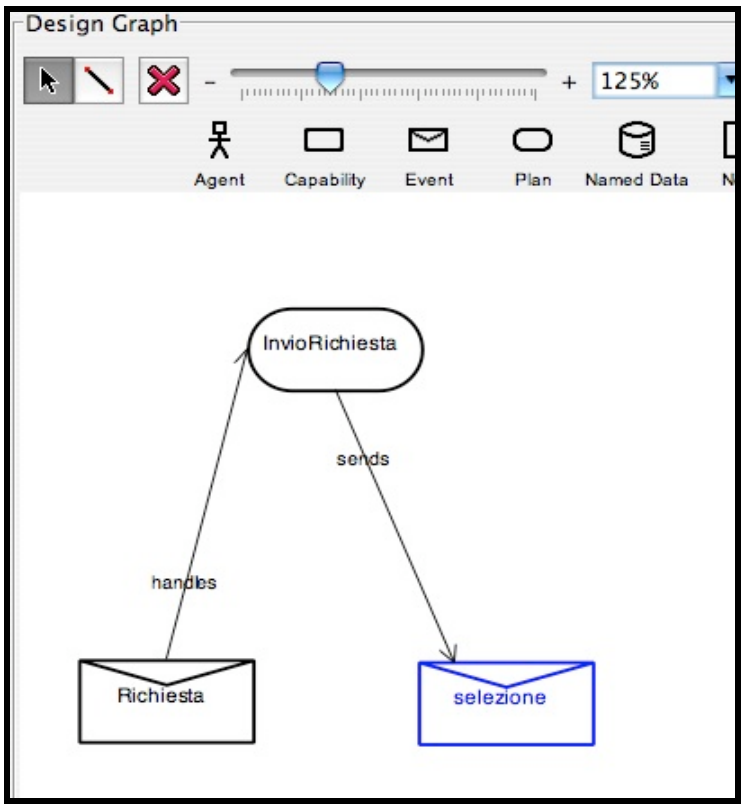
Analisi del sistema:

1. Struttura del sistema in Jack

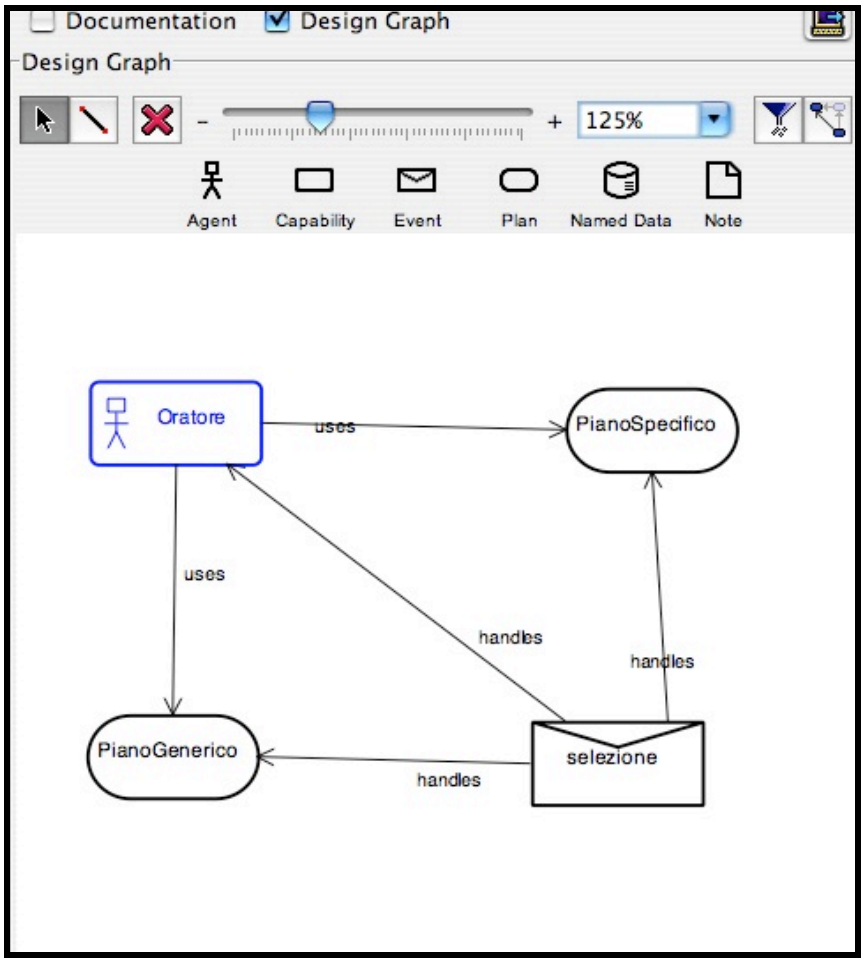
Nella piattaforma commerciale Jack si è sviluppato un sistema che tenesse conto sia dell'interazione utente sia della cooperazione e della coordinazione tra gli agenti. Si è articolato un MAS costituito da due agenti: Spalla e Oratore. Spalla si occupa di ricevere le richieste dell'utente e di passarle all'agente Oratore. L'agente Oratore quindi reciterà il poema richiesto.
Il sistema è stato sviluppato a partire dai tool grafici per il design,messi a disposizione da questa piattaforma.



Dall' immagine sopra, presa dai grafici di design di JACK, si vede che L'agente Spalla ha la capacità "Richiedi".



Dall'immagine sopra si nota che l'evento "Richiesta" (che scatta grazie alla capacità "Richiedi") gestisce il piano "InvioRichiesta" che lancia l'evento "selezione".



Dall'immagine sopra si nota che l'evento "selezione" viene gestito dall'agente Oratore attraverso i piani "PianoGenerico" e "PianoSpacifico" .
Si è quindi sviluppato un sistema in cui l'agente Oratore "conosce" l'Infinito di Giacomo Leopardi e l'inizio dell'Inferno di Dante Alighieri.
Ogni richiesta diversa da queste viene comunque gestita informando l'utente che non è possibile recitare qualcosa di diverso da Dante e Leopardi.

2. Struttura del sistema in TuCSoN

Il sistema sviluppato in TuCSoN è nato dall'intento di tradurre il sistema creato attraverso JACK in questa nuova infrastruttura. Considerando che entrambi consentono la programmazione degli agenti attraverso JAVA, il passaggio non è stato complicatissimo.
Si è perciò cercato di estendere questo sistema creato in JACK, intrucendo più agenti e quindi gestendo problematiche di interazione un po' più complesse.
Rispetto al precedente MAS si sono introdotte le seguenti "figure": Logger, Critico, Visualizzatore, Visualizzatore2.
L'agente Logger si occupa di memorizzare in un file html ogni richiesta che l'agente che l'agente Oratore riceve dall'agente Spalla. Si è introdotto questo componente perché così facendo si ha uno "storico" delle richieste. Potenzialmente tali file saranno riutilizzabili in futuro da un agente più complesso che crea delle statistiche sugli input utente.
L'agente Critico è un agente che può essere o meno utilizzato, a seconda delle richieste. Esso viene sfruttato dall'agente Spalla quando un utente desidera un commento a fianco del poema.
I due Agenti Visualizzatore e Visualizzatore2 si occupano di rappresentare attraverso una finestra grafica ciò che Oratore e Critico "producono" in risposta a ciò che l'agente Spalla gli richiede.

Realizzazioni

Il progetto del sistema in JACK è stato svolto a partire dai grafici, implementando alcune classi JAVA. Esso è utilizzabile da linea di commando o, più facilmente tramite il tool di JACK detto "Compiler Utility". Gli agenti Spalla e Oratore comunicano attraverso le primitive di JACK:

```
#reasoning method
body()
{
     System.out.println("Spalla: invio della richiesta");
     @send(pre.Oratore, post.parla(pre.Brano));
}
```

L’immagine sopra è presa dal piano “inviorichiesta” dell’agente Spalla e mostra come viene sfruttata la primitiva *@send*.

Il progetto del MAS in TuCSoN si è concretizzato creando degli agenti Java che dialogano attraverso un Tuple Centre “palco”. Analogamente a quando fatto in JACK, anche in questo caso si sono sfruttate delle funzioni proprie di tale infrastruttura di coordinazione per la interazione degli agenti. La stessa funzione vista nell’immagine sopra è stata riscritta come segue:

```
// creo un tuple centre
TupleCentreId tid = new TupleCentreId("palco@localhost");

// receiver of the message
Value dest = new Value("oratore");

// content of the message: in this case a simple string
Value msg = new Value(args[0]);


TucsonContext cnt = Tucson.enterDefaultContext();
cnt.out(tid,new LogicTuple("msg",dest,msg));
```

Non si è ritenuto necessario modificare il comportamento dei Tuple Centre con un opportuno file Respect. L’interazione tra gli agenti è piuttosto semplice in un contesto del genere.
Ed è anche per questo che si sono introdotti altri agenti rispetto al progetto realizzato in JACK. Così facendo si è potuto rifinire il sistema originario con delle finezze grafiche e con il log delle richieste.
Inoltre in fase di implementazione si è ritenuto corretto richiamare Visualizzatore e Visualizzatore2 attraverso due istanze rispettivamente in Oratore.java e Critico.java
Infine si è programmato l’agente Critico con un solo commento relativo al caso in cui l’utente richieda Leopardi, in quanto Dante è difficilmente riassumibile in poche parole.
Il sistema è stato sviluppato in modo “intelligente” in quanto se viene richiesto Dante e un commento, viene prodotto solo Dante e l’agente Critico non viene utilizzato.

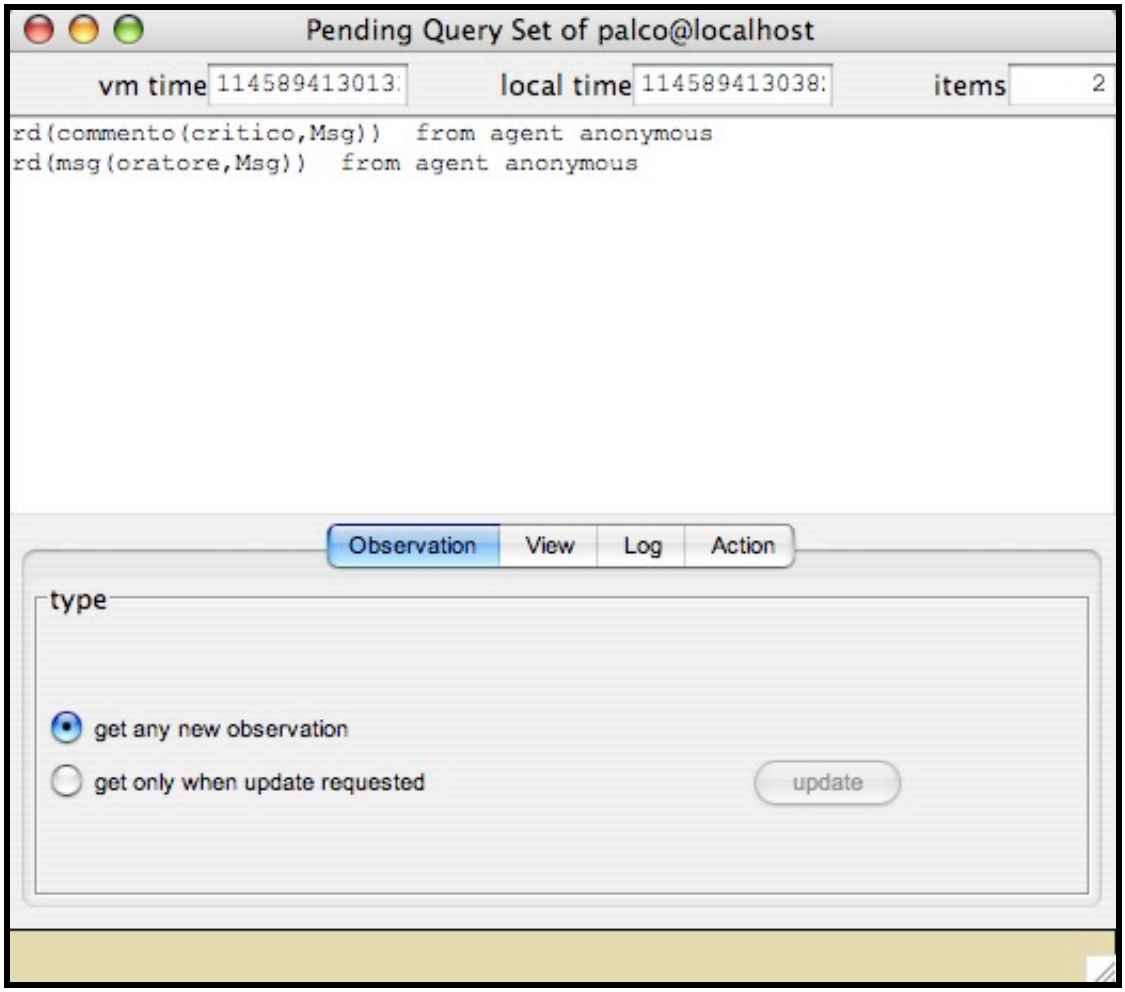
Ipotesi di sviluppi futuri

Il sistema MAS creato in TuCSoN contiene un agente "Logger" che a mio parere può avere un posto di rilievo nell'eventuale apporto di modifiche al sistema. Andando a leggere i file che tale agente "Logger" produce per ogni richiesta passata all'agente Oratore, si potrebbe realizzare un componente "Memory". Quest'ultimo potrebbe creare statistiche sulle richieste più probabili e passarle all'agente Oratore il quale potrebbe decidere di non rispondere ad un input perché ha già espletato troppe volte tale compito.
A mio parere sarebbe anche interessante consentire al sistema di comunicare con un DataBase. Si potrebbe quindi utilizzare le JDBC, che in pratica sono una API Java che permette, a partire da un qualsiasi programma scritto in java di accedere ad una sorgente di dati esterna: nella maggior parte dei casi un DBMS relazionale.
Il DB potrebbe essere sfruttato per consentire agli agenti di parlare in modo "criptato" (magari attraverso numeri), per evitare che la comunicazione sia chiara anche ad eventuali agenti esterni con fini pericolosi che sniffano dal centro di tuple utilizzato. In tale caso sarebbe sufficiente che l'agente Spalla richieda un poema all'agente Oratore attraverso una certa stringa, quest'ultimo interrogerà il DB e capirà la richiesta attraverso l'associazione "Stringa – Poema".

Confronto JACK - TuCSoN

Dopo aver sviluppato entrambi i sistemi posso dire che entrambe le tecnologie rappresentano soluzioni efficaci per la creazione di MAS ben strutturati. I tool grafici di JACK velocizzano e non di poco la scrittura degli agenti. Inoltre JACK rappresenta una vera e propria piattaforma completa per la progettazione, sviluppo e test delle proprie applicazioni in modo molto intuitivo. TuCSoN non rappresenta una piattaforma in senso classico ma è possibile scrivere le proprie applicazioni appoggiandosi ad un qualunque editor importando le opportune classi di TuCSoN. Inoltre la letteratura su TuCSoN è molto chiara e facilmente reperibile. TuCSoN inoltre risulta abbastanza facile da utilizzare e gli strumenti messi a

disposizione sono notevoli. L'inspector Tool di TuCSoN è davvero utile e consente innumerevoli funzioni che analizzano ogni cosa avvenga in un certo nodo con grande precisione (tuple presenti, reazioni del nodo agli eventi, query pendenti, stato della JVM).



Aggiungerei inoltre che il mio progetto non mi ha consentito di sfruttare appieno le funzionalità e le caratteristiche di TuCSoN perché non mi è stato necessario modificare il comportamento di nessun Tuple Centre per la realizzazione del mio sistema. Infatti tale possibilità offerta da TuCSoN, che si è sempre più orientato verso una intelligenza dei MAS concentrata nel "mezzo" piuttosto che nei singoli agenti, è uno degli elementi di spicco di tale infrastruttura.

A mio parere, una piattaforma in stile JACK con in aggiunta la possibilità di creare grafici in cui si settano in automatico anche i comportamenti dei TC e come gli agenti vi interagiscono,(con che primitive), sarebbe una soluzione ancora migliore delle due esistenti che faciliterebbe di molto la realizzazione di MAS.