Monitoraggio di MAS distribuiti attraverso grafica 3D

Relazione

Matteo Campana

Indice

Introduzione **3**
Aspetti Tecnologici **4**
2.1. Descrizione del sistema 4
2.2. Future Works 5
2.3. Concetti dei MAS attinenti al progetto 6

Capitolo 1

Introduzione

L'applicazione ha lo scopo di monitorare l'evoluzione di sistemi distribuiti intelligenti attraverso una rappresentazione grafica tridimensionale delle entità. Considerando sistemi MAS (Multi-Agent System) realizzati attraverso *Tuple Space model* (TuCSoN) diventa interessante la rappresentazione di Nodi, Tuple-Center e agenti. Il progetto nasce quindi dalla necessità di avere uno strumento che offra una rappresentazione semplice ed intuitiva delle varie entità, dalla quale ottenere utili informazioni sulla distribuzione, strutturazione ed evoluzione dei MAS esaminati. Per soddisfare questa necessità, l'utente ha la possibilità di accedere a tre livelli distinti:

- **Livello organizzazione:** monitoraggio di un insieme di nodi
- **Livello Nodo:** monitoraggio di uno specifico nodo (a questo livello si avrà visione di agenti e Tuple Center)
- **Livello Tuple:** monitoraggio di uno specifico agente o tuple center (visione di tuple)

Per descrivere l'evoluzione e il comportamento dei sistemi monitorati vengono sfruttate le proprietà di ambienti grafici 3D (colori, trasparenze, animazioni ecc…).

L'applicazione è a sua volta un MAS che utilizza TuCSoN come infrastruttura. Gli agenti sono realizzati in gran parte utilizzando il linguaggio Java. Attraverso java è infatti possibile accedere ai numerosi servizi offerti dalla libreria grafica Java3D con la quale modellare visivamente il sistema distribuito osservato. Per realizzare le funzioni di *governing ed enabling* del sistema si sono utilizzati agenti tuProlog.

Capitolo 2

Aspetti Tecnologici

In questo ultimo capitolo si analizzano le tecniche di progettazione e realizzazione del sistema. Nel primo paragrafo è descritta brevemente la struttura dell'applicazione mentre nel secondo vengono elencate alcune delle possibili modifiche e nuove soluzioni per ottenere una ottimizzazione del sistema. Infine verranno sottolineati alcuni aspetti degli agenti visti nei *sistemi intelligenti distribuiti* che durante la realizzazione del progetto sono stati sfruttati.

2.1. Descrizione del sistema

Lo scopo del sistema è chiaramente quello di ottenere informazioni dall'ambiente per poter poi elaborare un modello che ne rappresenti graficamente la distribuzione delle varie entità e il loro comportamento. Come già chiarito nei capitoli precedenti si è scelto di utilizzare TucSoN come infrastruttura. Una parte del lavoro è stata quindi dedicata alla realizzazione di una nuova versione di TuCSoN con un primo supporto all'osservabilità. L'idea di base è di utilizzare uno specifico Tuple Center ('$OBS') per ottenere una sorta di *logging* degli eventi che si verificano in tutti i nodi resi '*osservabili*'. Le informazioni raccolte saranno poi elaborate dall'applicazione grafica e trasformate in una rappresentazione visiva tridimensionale. Il progetto delle due parti è stato fatto da persone diverse; si è quindi resa prioritaria l'identificazione del formato delle tuple rappresentanti il verificarsi degli eventi.

Si vuole ora prendere in considerazione la sola parte grafica del progetto. Essa è costituita da un insieme di agenti realizzati in java. Uno degli agenti (*GraficHostAgent*) ha il compio di recuperare le informazioni utili a livello organizzazione: abilitazione e disabilitazione all'osservazione di nuovi nodi, elenco nodi presenti all'avvio dell'applicazione ecc…Il monitoraggio dei singoli nodi sarà invece ottenuto con altri agenti (genericamente chiamati *GraficAgTGAgent*) ognuno dei quali osserverà agenti e TC di un singolo nodo. Il sistema permette il monitoraggio contemporaneo di più nodi; per questo motivo non è possibile individuare a priori il numero di agenti che costituiscono il sistema.

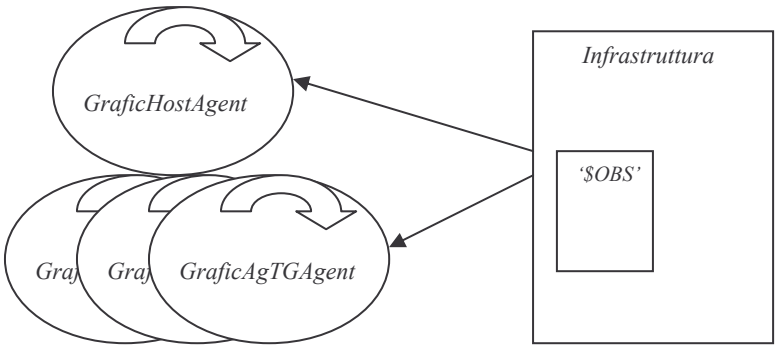

Figura 2.1

Le due tipologie di agenti sono state ottenute estendendo la classe *CadUnivers* che offre utili strumenti per creare e gestire ambienti tridimensionali (per una descrizione dettagliata di come si è realizzata la classe si guardi il manuale `CadUniverse.pdf`).

Questo fatto sottolinea che anche nella realizzazione di sistemi ad agenti, i vecchi paradigmi (in questo caso l'Object Oriented) mantengono le loro utilità e potenzialità.

In realtà il sistema è costituito da un ulteriore agente rispetto a quelli mostrati in fig. 4.1: l'agente mancante (*GovEnab*) è una semplice teoria tuProlog che si attiva solo al lancio dell'applicazione e ha la funzione di definire sia aspetti di *enabling* che di *governing*. In realtà possiamo considerare questa entità non come un agente ma parte integrante dell'infrastruttura.

Il file `theory.pl` e stato appositamente non compresso nel file `jar` per poterlo modificare a piacimento in base alle necessità o preferenze. Attraverso questa teoria è possibile ad esempio specificare regole comuni a tutti gli agenti; si crea cioè un sistema in grado di essere facilmente GOVERNATO da un entità 'centralizzata'.

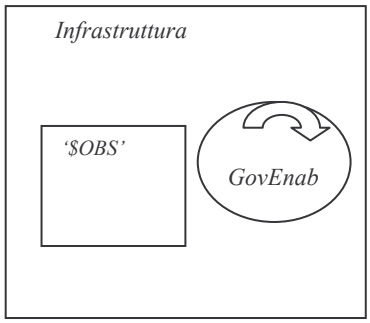

Figura 2.2

2.2. Future Works

Il sistema di monitoring non è molto complesso dal punto di vista di coordinazione tra le entità; l'unica complessità sta nella realizzazione di grafiche tridimensionali. L'implementazione poteva essere fatta anche senza utilizzare il concetto di MAS. Se si fosse optato per questo ultimo approccio si sarebbe però realizzato un sistema nettamente meno flessibile. L'utilizzo di un sistema ad agenti ha infatti da un lato semplificato l'individuazione e realizzazione di entità autonome, dall'altro offerto la possibilità di modificare e creare facilmente nuove funzionalità del monitoring.

Si pensi ad esempio alla possibilità di definire (attraverso il *GovEnab*) l'associazione dei colori ai comandi (nell'applicazione attuale i colori sono definiti internamente alle classi). Si potrebbe ad esempio definire la distanza spaziale dai TC alla quale gli agenti devono fermarsi; in questo modo si potrebbe modificare il parametro in base all'affollamento di agenti nei nodi. Si va cioè in una direzione in cui parametri comuni a molti agenti (colori, velocità, trasparenze, movimenti) sono gestibili da un'unica entità (*governing*).

Il sistema attuale permette al *GovEnab* di agire sull'infrastruttura solo durante il lancio del monitoring; si pensi ad uno scenario in cui l'azione del *GovEnab* sia continua e non limitata agli istanti iniziali; si pensi cioè alla possibilità di poter modificare parametri e comportamenti degli agenti a runtime (*online engeneering*).

Si potrebbe implementare un ulteriore agente con il compito di modificare tali parametri automaticamente in base alla numerosità e dinamicità del nodo monitorato (*percezione-azione*).

Il monitoraggio potrebbe essere esteso anche a sistemi che utilizzano tecnologie diverse da TuCSoN; l'idea potrebbe essere quella di sfruttare il TuCSoN come mediatore tra ontologie diverse (*Enabling interoperability*).

2.3. Concetti dei MAS attinenti al progetto

Cerchiamo ora di analizzare le caratteristiche delle entità presenti nel progetto, sfruttando i concetti visti durante il corso di *sistemi intelligenti distribuiti*.

Si è visto come il concetto di **agente** è differente in base all'area di studio; cerchiamo di elencare una serie di caratteristiche proprie degli agenti utilizzati nel monitoring:

- *Ha una rappresentazione del mondo*: percepisce il mondo leggendo le tuple contenute in '$OBS' e ne crea una rappresentazione grafica.
- *E' autonomo:* ogni agente ha un proprio flusso di controllo; non è possibile identificare un agente con un oggetto.
- *È situato nel mondo e vive in società:* situato in un mondo accessibile e dinamico.

Cerchiamo ora di analizzare l' *abstract architecture* degli agenti. Essi possono essere classificati come Agenti con stato: hanno infatti strutture di dati interne (liste di agenti e TC presenti ecc…) che permettono di memorizzare informazioni relative allo stato e storia dell'ambiente.

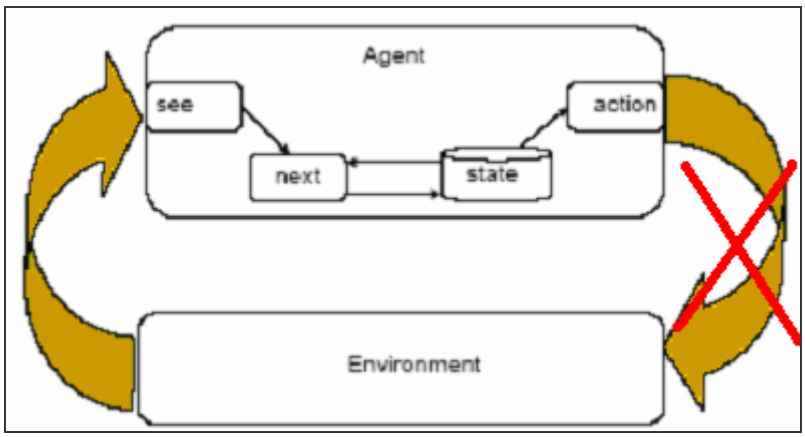

Figura 2.3

E' ovvio che la parte di azioni sull'ambiente non è molto importante negli agenti di monitoring; la loro funzione principale è osservare l'ambiente e produrne una rappresentazione (prima attraverso strutture dati e poi graficamente).

Rimane ora da capire quale è l'*architettura* degli agenti: *simbolici o reattivi*?!

Si dice che un agente o la sua architettura è deliberativa se contiene un esplicito modello che ne rappresenti il mondo…ok è deliberativo!!

Ma se vero che i sistemi reattivi sono quelli che mantengono continuamente una interazione con l'ambiente e rispondono ai cambiamenti o eventi che si verificano, possiamo considerare il sistema anche reattivo. Infatti il sistema rimane in ascolto di eventuali nuovi eventi notificati nel TC '$OBS' e reagiscono al verificarsi di questi.

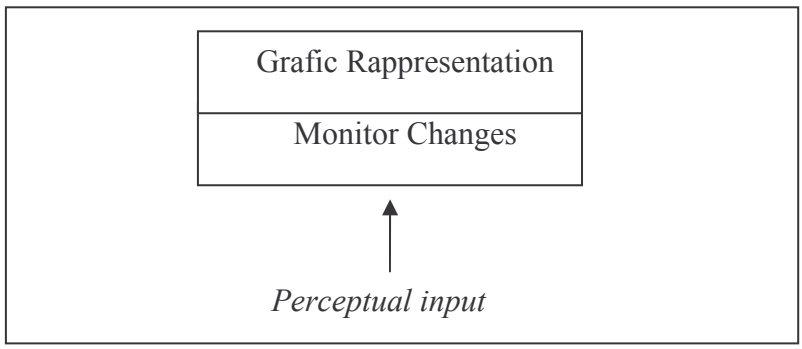

Figura 2.4

L'architettura (vedi fig. 4.4) è quindi ibrida, costituita di una parte reattiva che si occupa di segnalare il verificarsi di cambiamenti nell'ambiente e una parte deliberativa che ha il compito di realizzare e mantenere aggiornata la rappresentazione grafica.