ALINAs: un'architettura multi-layer ad agenti per il supporto alla comunicazione linguistica

M.T. Pazienza, A. Stellato, M. Vindigni,
DISP, Università di Roma "Tor Vergata"
{pazienza,stellato,vindigni}@info.uniroma2.it

Abstract—**Si presenta nel seguito ALINAs (Architecture for LINguistic Agents): una architettura multi-layer ad agenti specializzata al supporto della comunicazione linguistica.**

Essa si compone di una infrastruttura nella quale sono implementati i meccanismi di Percezione-Ragionamento-Azione (PRA) di base oltre al supporto per la comunicazione comprendente i performativi più comuni ai diversi linguaggi di comunicazione tra agenti (ACL), e di un layer specifico dedicato alla comunicazione linguistica.

Scopo dell'architettura è semplificare la sperimentazione di meccanismi di reasoning e di comunicazione tra agenti, svincolando il ricercatore dai problemi implementativi che altre architetture richiederebbero. La progettazione a componenti, l'alta modularità e il forte disaccoppiamento della sovrastruttura linguistica rende inoltre quest'ultima adattabile a diverse architetture di agenti.

I. Introduzione

La definizione di ambienti che consentano una migliore interoperabilità e modularità delle componenti è un obiettivo della comunità di ingegneria del software nella realizzazione di applicazioni software su larga scala. Motivo ricorrente nelle soluzioni proposte é il trasferimento del carico dell'interoperazione dagli sviluppatori e dagli utenti ai programmi stessi.

Diversi studi nell'ambito della Condivisione della Conoscenza suggeriscono la possibilità di estendere significativamente la tecnologia dell'Ingegneria del Software attraverso lo sviluppo di Agenti Software in grado di interagire tramite un linguaggio standard espressivo (Agent Communication Language, o brevemente, ACL). In questo contesto, gli agenti software sono stati proposti come uno strumento di supporto per affrontare il crescente volume e la complessità sia delle informazioni che delle risorse computazionali. In questo scenario gli agenti lavoreranno congiuntamente con degli strumenti per la gestione dei documenti per selezionare i dati corretti, assemblare le componenti necessarie e presentare l'informazione nel modo più appropriato per un utente e una situazione specifica.

Nonostante i diversi approcci alla tecnologia ad agenti appaiano significativamente promettenti, restano irrisolte varie questioni critiche, perché ancora gli agenti creati all'interno di un determinato *framework* possono raramente comunicare con gli agenti creati in un altro scambiandosi "conoscenze".

Molte delle architetture ad agenti esistenti (Jade, [10], [12], OAA, [9], [14] etc..), forniscono delle linee guida per la progettazione ad un livello di astrazione molto elevato, che non si dimostra abbastanza informativo da permettere di strutturare sistemi che operino efficientemente in ambiti altamente eterogenei. Ciò ha costituito la motivazione principale per il progetto di una architettura ad agenti a supporto della comunicazione linguistica, considerando il linguaggio come supporto per la rappresentazione e condivisione delle conoscenze.

II. Il problema della Condivisione della Conoscenza

Nel 1990, DARPA (Defense Advanced Research Projects Agency), ha dato vita al progetto KSE (Knowledge Sharing Effort: [5],[7]), con l'obiettivo di sviluppare tecnologie, metodologie e strumenti informatici per la condivisione dell'informazione.

Il concetto alla base del KSE è imperniato su una affermazione importante:

La condivisione della conoscenza richiede la presenza di un supporto comunicativo, e quest'ultimo, a sua volta, richiede un linguaggio comune.

Sebbene non direttamente parte del progetto originale KSE, il concetto di agente, quale entità appartenente a scenari eterogenei, incarna il problema della condivisione delle informazioni e di una struttura linguistica funzionale alla comunicazione.

Gli approcci a tale problema proposti in letteratura possono suddividersi in due aree principali:

- La ricerca di uno *standard*, un linguaggio condiviso da ogni agente che desideri poter comunicare con altri al di fuori del suo specifico ambiente.
- La *mediazione ontologica*, che prevede una esplicita traduzione dei concetti tra diverse ontologie.

Per quanto concerne lo standard, una prima proposta significativa è rappresentata da KIF (Knowledge Interchange

Format, [1] e [3]), un linguaggio logico proposto come *interlingua*, cioè un linguaggio universale che possa essere adottato da ogni agente. Questo approccio è stato oggetto di diverse critiche ([2],[5]), a causa della sua difficile realizzabilità e dei limiti intrinseci che esso impone (determinare uno standard specifico equivale a impedire il proliferare di altri linguaggi più utili in altri ambiti) , rappresentando più un valido terreno di ricerca che uno standard a tutti gli effetti.

Riguardo la mediazione ontologica, questa può a sua volta basarsi su diverse assunzioni circa il grado di condivisione delle conoscenze dei diversi agenti e il livello di eterogeneità delle comunità di cui fanno parte.

L'approccio "a priori" prevede la figura di un agente mediatore che operi la traduzione "on the fly" dei contenuti che due o più agenti desiderano scambiarsi (in [13] un possibile scenario, nel quale sono previsti, oltre al mediatore, un coordinatore, preposto al rendere possibile la comunicazione linguistica tra diversi agenti, e un eventuale traduttore dedito alla traduzione tra le diverse terminologie usate in una comunicazione).

L'estremo opposto, sempre in ambito di mediazione ontologica, è costituito dalle *mapping ontologies* ([6]) in cui si suppone un controllo totale sulle ontologie adottate: per ogni ontologia adottata nell'ambito di una comunità, devono essere realizzate delle operazioni di mapping tra essa e tutte (o alcune) le diverse ontologie previste nello scenario. Questo tipo di approccio, oltre a richiedere un forte accordo a priori tra le diverse parti della comunità (ogni ontologia viene registrata presso di essa) implica inoltre un effort notevole speso nella traduzione tra le diverse ontologie: d'altro canto ciò fornisce un solido sostegno per veicolare correttamente la semantica dei contenuti nella comunicazione tra agenti dotati di conoscenze eterogenee.

Quale sia la soluzione adottata, appare evidente come il compito di gestire la complessità dell'atto comunicativo non possa essere demandato agli stessi agenti comunicanti, ma a specifiche figure dedicate al supporto alla comunicazione.

III. ALINAS: UN PROTOTIPO DI ARCHITETTURA APERTA DI AGENTI PER IL SUPPORTO ALLA COMUNICAZIONE

Appare difficile realizzare un paradigma di agente generico che possa coprire in modo efficiente un ampio spettro di compiti differenti, come: l'interazione con l'utente, la gestione delle risorse informative, i processi decisionali richiesti dall'attività comunicativa e la conoscenza dei linguaggi.

Nelle applicazioni reali, sembra emergere una distinzione naturale tra risorse informative e componenti decisionali. Analogamente, in un'architettura multi agente si preferisce delegare i diversi tipi di responsabilità, quali garantire l'accesso alle informazioni e fornire la conoscenza di supporto per la gestione di compiti specifici, a tipi differenti di agenti software, essendo questa differenziazione utile anche per la comprensione delle dinamiche del sistema.

Inoltre, dal punto di vista dell'Ingegneria del Software, poiché si realizzano molteplici passaggi intermedi tra la produzione dell'informazione e il suo effettivo consumo, assegnare le varie fasi di processamento a componenti software separate può utilmente aumentare la modularità e la riusabilità delle componenti, anche in considerazione della possibilità che i risultati intermedi possano essere riutilizzati in altre interazioni, con entità diverse o in momenti successivi.

Con questi obiettivi, già in [13] erano state proposte tre differenti classi di agenti intelligenti, ognuna esibente caratteristiche progettuali peculiari orientate allo specifico compito cui è dedicata:

1. **agenti di risorsa**, che rappresentano gli estremi del processo comunicativo e possiedono la conoscenza oggetto di trasmissione;
2. **agenti di servizio**, che forniscono le singole funzionalità di supporto per lo svolgimento di attività complesse;
3. **agenti di controllo**, che forniscono la conoscenza strutturale del problema e possiedono abilità di coordinamento e di controllo.

La decisione progettuale di differenziare queste categorie è scaturita dalle seguenti osservazioni:

- Poiché uno degli usi primari degli agenti di risorsa é di rendere le risorse informative disponibili alla comunità di agenti interagenti, la tipizzazione presentata fornisce uno strumento di astrazione orientato a garantire lo sviluppo incrementale dei sistemi, in cui i progettisti possono focalizzare l'attenzione inizialmente solo su agenti specifici di risorsa e quindi cominciare ad utilizzare l'informazione disponibile prima di realizzare componenti per una sua gestione più avanzata.

- La suddivisione delle competenze tra agenti di servizio e agenti di controllo realizza la separazione tra cognizioni dei meccanismi "tattici" di risoluzione dei singoli problemi, e conoscenze "strategiche" di come comporli per la realizzazione di un obiettivo complesso. In tal modo, é possibile analizzare nuove problematiche, mediante una ricombinazione degli strumenti esistenti, e dualmente, potenziare le singole funzionalità senza dover necessariamente ristrutturare l'intero sistema.

Uno specifico agente può esibire una molteplicità di comportamenti che riflettono in diversa misura le caratteristiche peculiari delle singole classi: per esempio, qualunque risorsa é descritta come un agente con possibilità cognitive limitate a saper ricevere una richiesta puntuale ed a porre in essere le sole azioni necessarie a soddisfarla, mentre a

più alti livelli cognitivi sono posti agenti con ruoli di servizio. Questi ruoli, a seconda del livello di strutturazione scelto, possono essere realizzati anche dagli stessi agenti, dotati di capacità superiori; la tipizzazione cui si fa riferimento va quindi intesa per i soli scopi modellistici. E' utile a nostro avviso ulteriormente strutturare tali classi in livelli più o meno cognitivi, nell'ottica sia di specializzare le tipologie di agenti coinvolti, che di completare la modellizzazione dell'ambiente comunicativo.

Per tale motivo l'architettura proposta in questo articolo è basata su due differenti layer attraverso i quali strutturare e specificare le capacità dei diversi agenti. Per il livello inferiore sono state realizzate le capacità attive/reattive fondamentali di cui ogni agente è dotato, oltre al protocollo di comunicazione di base. Ad un livello immediatamente superiore sono implementate le funzionalità caratteristiche di ogni agente linguistico. Da quest'ultima classe di agenti, attraverso un ulteriore specializzazione, vengono generate le diverse classi di agente linguistici. I metodi e le funzionalità del layer superiore sono sufficientemente disaccoppiate da quello inferiore in modo da poter usare tale supporto linguistico anche in altre architetture ad agenti (come JADE [10], [12], o OAA, [9], [14]), ciò rende il modello portabile su diverse piattaforme ed applicazioni.

Una visione d'insieme dei costituenti principali dell'architettura ALINAs è descritta nel class diagram di Fig.1:

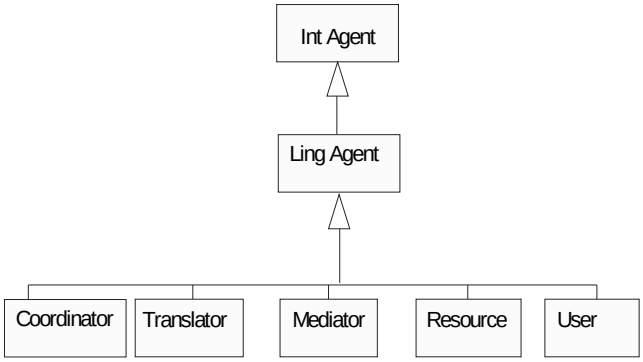

Figura 1. Class Diagram di ALINAs

ALINAs include le tre classi di agenti precedentemente descritte e, basandosi su questa prima categorizzazione, il layer linguistico fornisce ulteriori sottoclassi di specializzazione orientate al supporto alla comunicazione linguistica.

Nella fattispecie è prevista una particolare figura di agente di controllo, denominato *Coordinatore Linguistico*, il cui compito consiste nel provvedere alla gestione di una comunicazione linguistica tra due agenti che non siano in grado di condurla autonomamente (vedi interazioni nella seguente Fig. 2) .

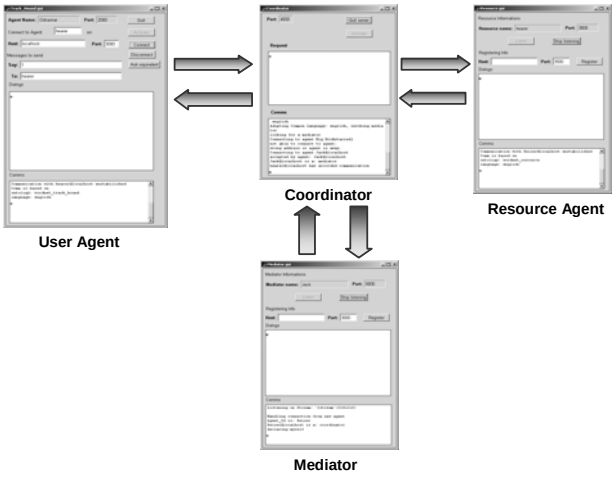

Figura 2. Un diagramma di interazione che mette in risalto il ruolo centrale che il coordinatore riveste nel supporto alla comunicazione linguistica.

I servizi di supporto alla comunicazione linguistica. per i quali sono previste opportune figure specializzate di agenti di servizio (Fig. 1) sono essenzialmente due:

- **Mediazione Ontologica,** indipendentemente dalla prescelta strategia di mediazione - uso di mapping ontologies o di mediazione in tempo reale - l'agente *Mediatore* provvede a fornire un ponte (vedi ancora interazioni in Fig.2) tra gli elementi di due ontologie, che siano legati da affinità concettuali ([13] e [11]).

- **Traduzione Linguistica,** servizio di traduzione tra due linguaggi differenti, fornito da uno specifico agente di servizio, chiamato *Traduttore*.

IV. IL PROTOCOLLO DI COMUNICAZIONE

Affinché agenti con differenti conoscenze ontologiche o linguistiche possano almeno intraprendere un tentativo di comunicazione, è necessario che vi sia un insieme minimo di primitive universalmente riconosciute nella comunità: tali primitive costituiscono il protocollo di comunicazione base usato e riconosciuto da tutti gli agenti afferenti.

La comunicazione tra agenti coinvolge più livelli: 1) un substrato che sottintende alla trasmissione vera e propria dei messaggi, come ad esempio il protocollo TCP/IP; 2), ad un livello più alto un protocollo di comunicazione tra agenti, cioè l'ACL condiviso dagli agenti presenti nella comunità; 3) uno scambio di informazioni più evoluto, basato sulle conoscenze del mondo di ciascun agente.

L'interfaccia dei vari agenti con il livello dedito al trasporto dei messaggi è responsabile inoltre di rilevare le connessioni in atto con altri agenti, la consegna dei messaggi e il monitoraggio del percorso che questi fanno nella rete per giungere a destinazione.

Gli agenti preparano quindi dei messaggi nel loro

linguaggio di comunicazione, ignorando completamente ciò che avviene al livello sottostante o il modo con cui tali messaggi sono fisicamente spediti.

L'indipendenza tra i due livelli è stata progettata per garantire la possibilità di usare la medesima piattaforma ad agenti anche con protocolli di trasporto dei messaggi differenti, cambiando esclusivamente l'interfaccia tra i canali di ingresso/uscita dell'agente e il protocollo di trasporto usato nell'ambiente in cui gli agenti operano.

Il linguaggio ACL di base prevede primitive di comunicazione per la connessione e disconnessione tra agenti, per la trasmissione dei messaggi in uscita e la gestione di quelli in ingresso.

Al momento di stabilire una connessione, ogni agente dichiara le proprie generalità all'agente con il quale desidera comunicare: queste consistono nel numero della porta sulla quale l'agente è in ascolto, e in una coppia di valori [Agent_Id, Agent_Host], tale coppia fornisce un nome unico per un agente. Delegando all'utente proprietario di un determinato indirizzo IP, il compito di usare nomi univoci per ogni agente residente su quel determinato host, viene garantita la univocità dei nomi nel sistema senza il bisogno di un ulteriore accordo a priori. Nello stesso ambiente possono quindi coesistere una molteplicità di agenti dotati del medesimo identificativo locale, purché risiedano su indirizzi differenti.

Quando un agente riceve una proposta di connessione, egli spedisce indietro una risposta, (*connect_reply()*), nella quale riferisce

- se ha accettato: (*connect_reply(accepted)*)
- o meno: (*connect_reply(rejected())*)

la proposta.

In caso di risposta negativa, l'agente che ha negato la sua disponibilità ha la facoltà (a seconda del suo atteggiamento e delle sue capacità) di specificare ulteriormente il motivo del rifiuto con il messaggio *connect_reply(rejected("motivo_del_rifiuto"))*, così che l'altro agente possa regolarsi di conseguenza.

Ecco alcuni esempi di rifiuto di una proposta di connessione:

rejected(busy)*:* l'agente è occupato e non può rispondere al momento.

rejected(busy(time))*:* l'agente è occupato per *time* secondi, dopodiché tornerà in funzione.

rejected(moved_to(new_address))*:* l'agente ha cambiato indirizzo, in tal caso la risposta proviene da un agente maggiordomo che comunica il nuovo indirizzo dell'agente desiderato.

Il protocollo di gestione delle connessioni non specifica quali comportamenti un agente debba intraprendere a seguito di una risposta negativa e della relativa motivazione acclusa. Esso si limita a indicare quali risposte possano essere fornite e con quale sintassi. Sarà poi il singolo agente, in base al suo livello cognitivo e al suo atteggiamento, a considerare in forma più o meno completa l'informazione pervenutagli per determinare il suo successivo atteggiamento.

Una volta che due agenti abbiano stabilito una connessione, essi possono attivare una comunicazione. Il primo passo fondamentale è la presentazione:

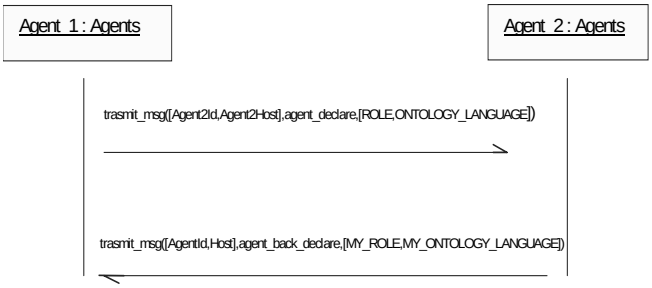

Figura 3: Presentazione tra due diversi agenti

Come è possibile osservare in figura 4, i due agenti si scambiano reciprocamente informazioni circa il loro ruolo all'interno del sistema; nel caso si tratti di agenti linguistici, essi forniranno come ulteriore argomento l'ontologia e il linguaggio che usano per comunicare.

A questo punto entrambi gli agenti conoscono tutto ciò che c'è da sapere sull'altro e, in particolare, l'agente che ha deciso di intraprendere una comunicazione può ora prendere delle decisioni sulla base delle informazioni recepite.

Nello scenario proposto coesistono agenti in grado di intraprendere autonomamente una comunicazione (ammesso di avere ontologie compatibili con gli agenti con cui entrano in contatto) e altri che si affidano completamente all'aiuto di un coordinatore, indipendentemente dalle proprie conoscenze ontologico/linguistiche.

Stabilita una comunicazione, il generico agente ha a disposizione performativi come ask e ask_many (e i relativi messaggi di risposta) per dialogare con altri agenti e ottenere informazioni da essi; ogni atto comunicativo è contraddistinto da un campo Subject_Id, cioè un codice che distingua ogni richiesta da altre analoghe che possono essere effettuate dallo stesso agente.

È previsto infatti che all'interno di una singola comunicazione tra due agenti, ciascuno possa indirizzare all'altro più richieste, le cui risposte potrebbero tornare in modo asincrono, il campo Subject_Id permette quindi di distinguere tali risposte, riordinarle e riprendere quindi la gestione della comunicazione.

TABELLA 1. PERFORMATIVI PIÙ COMUNI DEL LINGUAGGIO ALINAS' ACL

ask(Subject_Id,Question,Args) il performativo più semplice per ottenere delle informazioni da un agente; il secondo parametro prevede una richiesta specifica mentre il terzo gli eventuali argomenti di tale richiesta
ask_to(Agent,Subject_Id,Question,Args) permette di chiedere ad un agente di inoltrare una richiesta ad un terzo agente
ask_many(Subject_Id, Questions, Args) permette di spedire più richieste ad uno stesso agente
ask_many_to(Agent, Subject_Id, Questions, Args) permette di inoltrare una messaggio con richieste multiple (*ask_many)* a un terzo agente
Answer(Subject_Id,Question.Args) Risposta ad una *ask*
answer_many(Subject_Id, Questions, Args) Risposta ad una *ask_many*
answer_from(Agent,Subject_Id,Question,Args) Risposta ad una *ask_to*
answer_many_from(Agent, Subject_Id, Questions, Args) Risposta ad una *ask_many_to*

In genere il performativo *ask_to* è indirizzato agli agenti di controllo, ed è usato, qualora un agente di questo tipo stia coordinando una comunicazione, dai due agenti coinvolti nella stessa. In questo caso, un agente spedisce una richiesta ad un agente di controllo per avere informazioni da un determinato agente con il quale è in comunicazione, l'agente di controllo spedisce la richiesta all'agente interessato, questi risponde con una *answer* semplice, e successivamente l'agente di controllo inoltra la risposta all'agente richiedente mediante una *answer_from*.

Non è necessario spedire più messaggi di richiesta ad uno stesso agente se le richieste in essi presenti sono indipendenti tra loro (la formulazione di una richiesta non dipende dalla risposta ottenuta su un'altra richiesta). *Ask_many* permette quindi di snellire l'overhead della comunicazione riunendo più richieste in un unico messaggio.

Come accennato in precedenza, gli agenti linguistici hanno però modi ancora più evoluti di comunicare, la cui semantica è descritta nelle loro ontologie. Tutti gli agenti dotati della sovrastruttura linguistica definita in ALINAs, sono in grado di rispondere a delle richieste specifiche come il linguaggio che usano per comunicare o la particolare ontologia di cui sono dotati (qualora questa sia di un tipo riconosciuto all'interno della comunità), possono quindi accordare modalità comunicative con altri agenti o delegare tale compito ad agenti controllori linguistici che provvederanno ad inoltrare le richieste relative o nel caso ad invocare l'aiuto di ulteriori figure di supporto alla comunicazione linguistica (Mediatore e Traduttore).

TABELLA 2. ALCUNI DEI PERFORMATIVI ORIENTATI ALLA COMUNICAZIONE LINGUISTICA

say(message) è un performativo "general purpose" dedicato alla trasmissione di informazioni su base linguistica. La comprensione delle semantiche del messaggio in esso contenuto è completamente demandata alle capacità interpretative dell'agente
accord_communication(Ontology,Language) permette ad un agente (o ad un coordinatore per conto di un agente) di accordare le modalità linguistico/ontologiche sulle quali basare una comunicazione.
communication_accorded(Ontology,Language) è il messaggio rilasciato da un agente per indicare la sua disponibilità a intraprendere una comunicazione linguistica basata sulla ontologia e sul linguaggio proposti.
comm_request(DestAgentId, DestHost, DestPort) questo performativo è tipicamente inoltrato ad un coordinatore linguistico al fine di ottenere una comunicazione con un terzo agente
mediate_concept(Hearer,Concept) anche questo performativo è indirizzato ad un agente controllore linguistico, al fine di ottenere una mediazione ontologica circa il concetto che si vuole esprimere. Il coordinatore, che ha in memoria i dati riguardanti le comunicazioni che sta gestendo e gli agenti in esse coinvolti, invoca l'aiuto di un mediatore ontologico chiedendo
find_Equivalent_Concept(Speaker, Hearer, Concept) questo messaggio è tipicamente spedito da un coordinatore linguistico ad un agente mediatore, al fine di ottenere una mediazione ontologica[1] del concetto tra i due agenti.

V. ARCHITETTURA DEL SISTEMA

L'architettura ALINAs è costruita su una serie di livelli che realizzano le varie procedure di Percezione, Ragionamento e Azione.

Questa strutturazione, pur nella sua specificità per il supporto della comunicazione linguistica, rientra nella ampiamente condivisa letteratura delle architetture ad agenti.

I livelli che costituiscono tale architettura sono:

Il TCP Layer, ovvero il protocollo di trasporto messaggi

Il Base Protocol Layer (BPL), che fornisce il substrato comune per la gestione degli eventi generatisi nel sistema/o nella rete

Il Message Layer (ML), che si occupa della compilazione/interpretazione dei messaggi provenienti da o

[1] Per i dettagli sui processi di mediazione ontologica, vedi [13] e [11]

diretti a gli altri agenti

Il percept/React Layer (PRL), che realizza il livello reattivo dell'architettura, in cui l'agente risponde direttamente agli stimoli percepiti, eventualmente consultando la base di conoscenza

Il Reasoning Layer (RL), che rappresenta il livello in cui si realizzano i compiti più complessi, non direttamente gestibili in termini di stimolo/risposta e l'interazione con la base di conoscenza

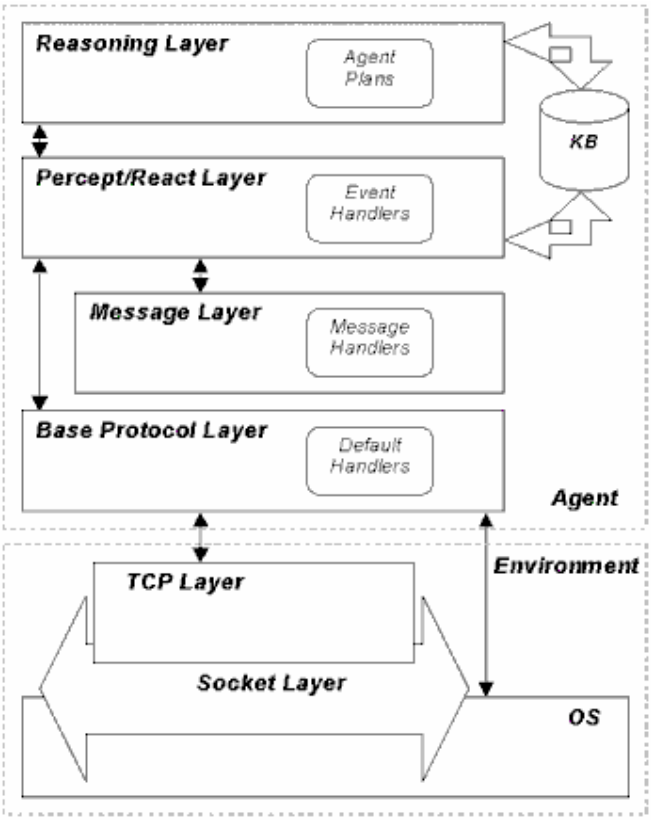

Figura 4. Architettura del sistema

Non tutti gli strati definiti sono necessariamente presenti nei vari agenti, come non è detto che tali strati vengano tutti usati in una singola operazione. Agenti di servizio e di controllo possiedono generalmente tutti i livelli, e fanno un uso effettivo sopratutto del livello di reasoning per la gestione delle situazioni specifiche per cui essi sono invocati. Al contrario, agenti di risorsa possono o meno presentare il livello di reasoning, a secondo dell'intelligenza richiesta. Agenti di risorsa più semplici, che svolgono un ruolo essenzialmente passivo nel sistema fornendo esclusivamente un'interfaccia per la consultazione dei dati sottostanti, sono usualmente privi di tale livello, e interagiscono solo in termini di risposta diretta alle interrogazioni esterna. Il livello percettivo e reattivo sarà più o meno sviluppato, a seconda della complessità degli stimoli che devono essere gestiti, riducendosi notevolmente nel caso in cui gli eventi percepibili siano standardizzati per l'intero sistema, nel qual caso essi saranno direttamente gestiti nello strato di base.

Il livello TCP/IP fornisce i servizi di basso livello per l'accesso alla rete, consentendo di creare e monitorare connessioni tra le macchine tramite l'uso di *stream sockets*. Le *stream sockets* sono il tipo più comunemente utilizzato di socket, che fornisce un flusso di dati bidirezionale e affidabile tra i programmi connessi.

A. Il Base Protocol Layer

Il Base Protocol Layer realizza il livello di interfaccia tra l'implementazione specifica dell'agente ed il sistema e contiene le procedure di basso livello che devono essere condivise tra tutti gli agenti, come ad esempio i dettagli delle connessioni TCP. Esso ha il compito di tradurre i segnali provenienti dall'ambiente (il sistema operativo o la rete) in informazioni gestibili dai livelli superiori. Eventi tipici come la sospensione dell'agente, la terminazione dell'agente, la presenza di un pacchetto di informazioni su una connessione aperta, la richiesta di connessione sono gestiti da procedure di default contenute in questo livello, qualora in fase di creazione dell'agente non sia stata dichiarata la presenza di un handler specifico nel livello superiore. In questo modo é possibile evitare la propagazione ai livelli superiori di quelle situazioni di routine il cui comportamento può essere definito "a priori" (come i messaggi di "connessione rifiutata" qualora l'agente sia non attivo). Questo livello si occupa di gestire gli eventi riguardanti l'esterno generati dall'agente, di inviare i messaggi nella rete e di gestire le operazioni necessarie alla interazione con il sistema operativo e il livello TCP.

B. Il Percept/React Layer

Il Percept/React Layer si occupa della gestione degli eventi generati dall'agente o dall'esterno. Quando il PRL riceve il controllo esso attiva le opportune procedure di gestione (event handlers) dei vari eventi. Quando un messaggio giunge all'agente, o un qualsiasi evento avvenuto nel sistema è percepito dall'agente, quest'ultimo può reagire in due modi:

- riflesso condizionato
- avvio processo di reasoning

Nel primo caso, il messaggio ricevuto o l'evento percepito non sono tali da richiedere un ragionamento particolare, l'agente reagisce immediatamente allo stimolo senza passare il controllo al Reasoning Layer; la risposta determinata da tale reazione automatica può al limite dipendere dallo stato dell'agente o, come avviene di norma nel caso di agenti di risorsa, da una interrogazione di una base di dati sottostante.

Nel secondo caso l'agente passa il controllo ad un processo di reasoning che determinerà le future azioni dell'agente.

C. Il Message Layer

Il Message Layer si occupa dell'interpretazione dei messaggi ricevuti dall'agente e del confezionamento dei

messaggi dell'agente verso l'esterno. Poiché l'agente percepisce il mondo esterno in termini di eventi, in questo livello avviene l'opportuna conversione tra il linguaggio interno dell'agente, e il suo linguaggio di comunicazione. Esso viene attivato dal PRL, in seguito alla presenza di un evento di notifica di ricevimento di un messaggio, o in seguito alla generazione di un evento di comunicazione. Qualora un evento segnali la presenza di un messaggio, il PRL attiverà il message layer dove il messaggio viene prelevato dalla relativa coda (vedi meccanismi di I/O), viene interpretato e vengono generati gli opportuni eventi dell'agente. In questo modo i messaggi provenienti dagli altri agenti sono opportunamente tradotti in termini di relativi eventi che saranno processati dal livello di percept/react. Allo stesso modo, qualora si generi un evento di comunicazione, il controllo viene passato a questo livello, che preparerà il messaggio nel formato opportuno e genererà un evento di avviso al PRL informandolo della presenza di un messaggio che deve essere spedito. La presenza del message layer consente di disaccoppiare il livello di comunicazione dal livello di gestione interna, permettendo all'agente di separare il linguaggio di comunicazione condiviso da quello utilizzato per la sua rappresentazione interna.

D. Il Reasoning Layer

Qualora la situazione richieda un'elaborazione non direttamente gestibile dal livello di Percept/React, viene attivato il Reasoning Layer. In questo livello sono gestiti gli obiettivi "di lunga durata" che l'agente sta perseguendo (ad esempio la mediazione tra i concetti delle due ontologie). Quando un nuovo obiettivo viene generato, esso provvede a decomporlo in sottoobiettivi successivi che devono essere realizzati, continuando a svilupparli fino alla loro risoluzione o fino a quando essi non risultano bloccati per l'assenza di informazioni adeguate, nel qual caso viene generato un evento di richiesta per il Percept/React Layer, a cui viene ripassato il controllo; questo provvederà eventualmente a chiedere al Message Layer di spedire un messaggio a qualche agente per ottenere le informazioni desiderate, oltre che a preparare un percettore (vedi paragrafo sulla implementazione) che si desti all'arrivo di tali informazioni. Qualora in seguito l'informazione richiesta divenga disponibile (per l'arrivo di un messaggio dall'esterno in risposta ad un'interrogazione), il Reasoning Layer sarà informato dal livello sottostante e riprenderà l'elaborazione da dove era stata sospesa.

E. Meccanismi di I/O

Una volta attivato, un agente apre un canale di ingresso (utilizzando il meccanismo degli stream socket prima descritto) attraverso il quale può ricevere richieste di connessione, messaggi e tutti gli altri segnali provenienti dall'ambiente.

La ricezione dei messaggi avviene mediante scansione periodica del flusso (stream) associato al socket di ingresso. I messaggi vengono quindi processati mano a mano che l'agente li riceve sul suo flusso di ingresso, nel caso in cui più di uno arrivi contemporaneamente, si formerà una coda che andrà svuotata con il processamento dei messaggi.

Ogni qual volta un agente riceve un tentativo di connessione da un altro agente e decide di accettarlo, apre per lui un apposito canale di comunicazione, e ogni messaggio giunto tramite esso proverrà esclusivamente dall'agente cui il canale è stato assegnato.

A regime operativo ogni agente sarà contemporaneamente in ascolto su una lista di stream associati agli agenti con i quali si trova in contatto più un singolo stream usato per ricevere nuove richieste di connessione.

F. Message Processing

Nella implementazione presentata, i messaggi vengono processati serialmente, sia che vengano individuati uno alla volta sulla lista dei canali di ingresso, sia che giungano simultaneamente (in questo caso vengono processati in base ad un ordinamento assegnato ai vari strema: per esempio, l'ordine con il quale sono stati aperti).

In ogni caso, una volta processati i messaggi, le relative richieste contenute in essi entreranno in una coda e saranno esaudite con disciplina FIFO.

E' possibile pensare ad agenti con la capacità di eseguire più funzionalità in parallelo. La disciplina FIFO può essere sostituita da altre politiche di servizio più evolute (come quelle con priorità tempo o con priorità dipendente dal grado di importanza di chi richiede il servizio), volte a migliorare il tempo di servizio medio offerto dall'agente o la qualità dello stesso.

VI. RIFERIMENTI IMPLEMENTATIVI DI ALINAS

L'architettura proposta, in linea con gli obiettivi di semplicità e predisposizione alla sperimentazione cui era destinata, è stata realizzata interamente in linguaggio SWI-Prolog, un dialetto prolog (http://www.swi-prolog.org) utilizzato da una vastissima comunità di ricerca, aperto alle nuove esigenze delle tecnologie multimediali e alle novità della Semantic Web (handling di file XML, RDF, HTML, SGML) e dotato di interfacce verso i più comuni linguaggi di programmazione (C, C++, Java) e gestione di componenti (CORBA, OLE).

I meccanismi di percezione-azione sono stati implementati estendendo la libreria "broadcast" di XPCE, un plug-in di SWI-Prolog per la gestione di interfacce grafiche ed eventi di sistema (Fig. 6).

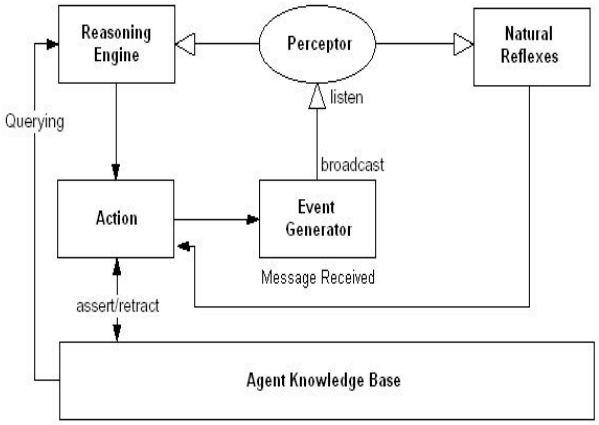

Figura 5: Ciclo PRA degli agenti ALINAs

VII. CONCLUSIONI

Punto focale dei paradigmi agent-oriented è la comunicazione. È attraverso di essa che gli agenti possono collaborare per giungere ad una soluzione dei problemi che si prefiggono di risolvere o agli obiettivi che sono loro assegnati: da qui il crescente interesse verso la standardizzazione di linguaggi che possano codificare la conoscenza posseduta dagli agenti.

L'introduzione di linguaggi formali per la rappresentazione della conoscenza (XML o RDF) ha costituito un valido contributo in tal senso; tuttavia, tali descrizioni sono limitate dalla mancanza di un ampio consenso sulle concettualizzazioni dei domini: se esiste quindi un accordo circa i possibili mezzi per rappresentare e condividere la conoscenza, non vi è invece uno standard sulle modalità di tale rappresentazione.

Ne emerge uno scenario eterogeneo di aspetti ontologici e linguistici, nel quale agenti diversi devono poter comunicare tra loro anche in assenza di un agreement ontologico sul livello di rappresentazione e/o di contenuti.

In questo scenario il linguaggio naturale acquisisce un'importanza rilevante per due motivi:

- è un mezzo universalmente adottato per la trasmissione dell'informazione,
- rappresenta un ulteriore avvicinamento delle macchine alle necessità dell'uomo, condividendone le capacità espressive e garantendo il loro uso da parte di fasce sempre più allargate di utenza.

Di converso, l'uso del linguaggio naturale come mezzo di comunicazione introduce un ulteriore livello di difficoltà di gestione legato alla sua intrinseca ambiguità espressiva. Le capacità necessarie per risolvere tali ambiguità lessicali comportano un alto grado di competenza linguistica. Sorge quindi la necessità delle figure specializzate e dedicate al supporto della comunicazione linguistica descritte e implementate.

In particolare, ALINAs si distingue da altre architetture esistenti in quanto rappresenta un prototipo di architettura ad agenti semplificata capace di svincolare la sperimentazione di meccanismi di reasoning e di comunicazione tra agenti dai problemi implementativi; per tale ragione essa si presta ad una diffusione in ambito scientifico e per fini didattici e, al tempo stesso, fornisce un valido e robusto mezzo di supporto ad applicazioni basate sulla comunicazione linguistica.

L'alta modularità e il forte disaccoppiamento di tale sovrastruttura linguistica permette inoltre il suo adattamento alle più complete e sofisticate architetture oggi disponibili.

VIII. Riferimenti

[1] M.R. Genesereth. *A KIF Proposal*. Technical Report, Stanford University, 1990.

[2] M. Ginsberg: *Knowledge Interchange Format: the kif of death*, AI Magazine, Vol.5, No.63, 1991

[3] M. R. Genesereth, R. E. Fikes, et al. *Knowledge Interchange Format Version 3 Reference Manual*, Logic-92-1, Stanford University Logic Group, 1992.

[4] R.S. Patil, R.E. Fikes, P.F. Patel Schneider, D. McKay, T. Finin, T.R. Gruber, and R. Neches.
The DARPA Knowledge Sharing Effort: Progress Report. In C. Rich, B. Nebel, and W. Swartout, editors, Proc. Of the Third International Conference on Principles of Knowledge Representation and Reasoning, Cambridge, MA, 1992, Morgan Kaufmann.

[5] A. E. Campbell, H. Chalupsky, and S. C. Shapiro. *Ontological Mediation: An Analysis*. Unpublished manuscript.

[6] P. Borst, J. Benjamins, B. Wielinga, H. Akkermans, *An Application of Ontology Construction*, Workshop on Ontological Engineering, ECAI'96, Budapest, PP. 5-16, 1996.

[7] R.S. Patil et al. *The DARPA Knowledge Sharing Effort: Progress Report*, Readings in Agents, M. Huhns and M. Singh, eds., Morgan Kaufmann, 1997.

[8] A.E. Campbell and S.C. Shapiro, *Algorithms for Ontological Mediation*, Technical Report 98-03, Dep. of CS and Engineering, State Univ. of New York at Buffalo, 1998.

[9] D. Martin, A. Cheyer, Moran, B. Douglas. *The Open Agent Architecture: A Framework for Building Distributed Software Systems*. Applied Artificial Intelligence, vol. 13, no. 1-2, pp. 91-128, January-March 1999.

[10] F. Bellifemine, A. Poggi, G. Rimassa, "JADE – A FIPA compliant Agent Framework," in *Proceedings of PAAM '99*, London, England, April 1999

[11] A. E. Campbell, *Ontological mediation: finding translations across dialects by asking questions*, PHD Thesis December 3, 1999

[12] F. Bellifemine, A. Poggi, G. Rimassa, P. Turci.
An Object Oriented Framework to Realize Agent Systems.
Proceedings of WOA2000 Workshop, Parma, May 2000, pagg. 52-57.

[13] M.T.Pazienza, M. Vindigni,
Language-based agent communication, Proceedings of the EKAW-02 workshop "OMAS Ontologies for Multi-Agent Systems", Siguenza (Spain) 30 sept. 2002

[14] A. Cheyer and D. Martin. *The Open Agent Architecture*. Journal of Autonomous Agents and Multi-Agent Systems, vol. 4 , no. 1, pp. 143-148, March 2001