Modellazione e Verifica di sistemi Multi-Agente Real-Time: Il Framework REMM

Aversa Rocco
Second University of Naples
rocco.aversa@unina2.it

Di Martino Beniamino
Second University of Naples
dimartino.beniamino@unina.it

Francesco Moscato
Second University of Naples
francesco.moscato@unina.it

Salvatore.Venticinque
Second University of Naples
salvatore.venticinque@unina2.it

ABSTRACT
I Sistemi Multi-Agente (MAS) rappresentano un nuovo mezzo per la modellazione di sistemi distribuiti complessi. Di recente si sono compiuti molti sforzi per definire metodologie e strumenti in grado si supportare la progettazione e lo sviluppo dei MAS e tuttora campo aperto di ricerca resta quello della verifica formale di questi sistemi, specialmente nel caso in cui questi sono sottoposti a vincoli real-time. Il framework presentato segue una metodologia per il progetto e lo sviluppo di MAS real-time, proponendo un ambiente integrato per il progetto, la generazione di codice e la verifica sia in fase di progetto che di esecuzione, di questi sistemi.

Keywords
Sistemi Multi-Agente, Model Checking, Real-Time, UML.

1.INTRODUZIONE
I sistemi multi-agente rappresentano una tecnologia relativamente nuova nel campo informatico, che ha destato un grande interesse negli ultimi anni grazie alla sua promessa di un nuovo paradigma per modellare, progettare e implementare complessi sistemi software [1,2]. Essi rappresentano un paradigma software appropriato per descrivere sistemi aperti e distribuiti, le cui caratteristiche cambiano continuamente nel tempo, e applicazioni real-time[5], in cui assumono sempre maggiore importanza sistemi software in grado di mostrare un comportamento flessibile, intelligente, in grado di adattarsi a condizioni di funzionamento differenti e non-deterministiche, con lo scopo di eseguire elaborazioni che siano prevedibili, rispettose dei vincoli temporali imposti.

Diverse metodologie sono state sviluppate per la modellazione di sistemi complessi secondo questo paradigma[3,4].

Il presente lavoro si inserisce nel contesto dei sistemi multi-agente operanti in tempo reale, con l'obiettivo di fornire un nuovo approccio allo sviluppo degli agenti in grado di seguirne l'intero ciclo di vita.

In particolare, il contributo originale del lavoro prende forma nella definizione di una metodologia per l'analisi, la progettazione e l'implementazione orientate agli agenti, che risponda alle seguenti esigenze:

- definire un modello logico con il quale rappresentare e progettare gli agenti software;
- individuare gli elementi necessari ad arricchire il modello con le informazioni che caratterizzano i sistemi real-time;
- effettuare una traduzione del modello verso una piattaforma ad agenti di riferimento, con conseguente generazione di codice;
- implementare un meccanismo di verifica dinamica dei vincoli real-time definiti per gli agenti.

I punti evidenziati costituiscono delle attività che nel loro insieme forniscono all'utente gli strumenti di supporto alla realizzazione di sistemi multi-agente. Per esse viene descritto un ciclo di sviluppo che ne mette in luce le operazioni fondamentali e il modo in cui esse sono state integrate tra di loro.

Sulla base di questa metodologia e del suo ciclo di sviluppo, pertanto, è stata progettata ed implementata l'architettura di un Framework che concretamente possa rispondere alle esigenze prima evidenziate.

Il Framework (REal-time Modeling of Mas: REMM) dispone di uno strumento per la modellazione del sistema multi-agente, con il quale descrivere mediante dei formalismi visuali la struttura degli agenti ed il loro comportamento sociale. In particolare, il modello adottato incorpora i principi propri della logica BDI (Beliefs, Desires, Intentions). Il sistema deve essere applicato ad un contesto real-time, in cui l'esecuzione degli agenti deve avvenire nel rispetto di determinati vincoli temporali. Da queste considerazioni è nata l'esigenza di estendere il modello BDI degli agenti con le informazioni necessarie ad implementarne l'elaborazione real-time.

REMM dispone di uno strumento di prototipazione in grado di generare uno scheletro di codice per ciascun agente. Lo strumento implementa un meccanismo di verifica della correttezza del modello degli agenti, con il quale determinare se il modello è compatibile con la piattaforma di esecuzione degli agenti presa in considerazione. All'interno del Framework REMM è incorporato, inoltre, uno strumento di verifica, che si integra con il codice dell'agente e ne esegue dinamicamente il controllo, invocando qualora necessario un model-checker per determinare se le proprietà temporali definite per gli agenti siano rispettate.

notare che questo potrebbe prevedere un cambiamento dei piani

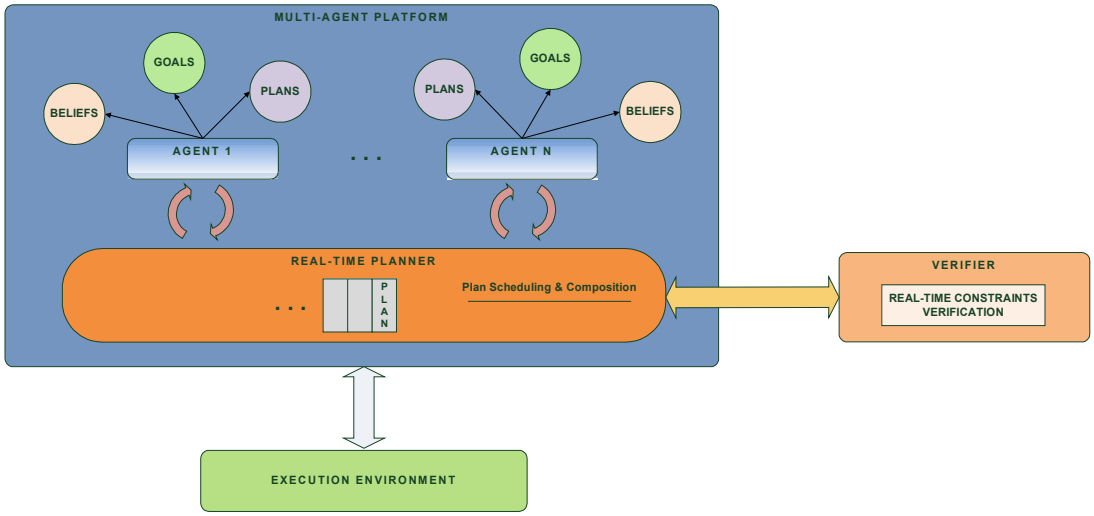

Figura 1: REMM: Architettura generale

2.ARCHITETTURA E CICLO DI SVILUPPO

Il sistema generale a cui si fa riferimento può essere rappresentato come in Figura 1.

Il modello rappresenta un sistema multi-agente operante in tempo reale, in cui gli agenti, caratterizzati da credenze, desideri ed intenzioni (gli elementi propri della logica BDI), sono gestiti all'interno di una piattaforma per agenti (Multi-agent Platform), ed operano su un campo applicativo definito dall'Execution Environment. In questo ambiente essi interagiscono e collaborano tra di loro attraverso le azioni che compongono i piani che hanno a disposizione. In particolare, essendo il contesto applicativo vincolato a dei requisiti temporali, diventa di fondamentale importanza che ogni piano venga eseguito nel rispetto degli stessi. La scelta dei piani da eseguire e la verifica dei vincoli real-time sono operazioni effettuate da due componenti, rispettivamente il Planner ed il Verifier.

Il Planner ha il compito di schedulare e comporre tra di loro i piani di ogni agente per far sì che questi possano raggiungere i propri obiettivi, anche attraverso "strade alternative". In questo contesto, infatti, si presenta il problema della ri-pianificazione, in quanto una sequenza di azioni o piani per la quale risulta impossibile conseguire il goal deve essere sostituita con una sequenza alternativa e presumibilmente decisiva per il raggiungimento del goal. Essendo l'elaborazione di tipo real-time, il Planner necessita di un meccanismo per stabilire se il piano in esecuzione è in grado di raggiungere l'obiettivo entro i tempi stabiliti (ad es. in fase di progetto). Questa verifica, eseguita a tempo di esecuzione, viene effettuata dal Verifier.

Per grandi linee, attualmente il planner sceglie (anche a run-time) tra i piani definiti nella fase di progetto quelli che sono in grado di raggiungere il goal preposto rispettando i vincoli real-time. A run.time, quando un piano fallisce nel raggiungere un goal, i piani della fase di progetto vengono ricanalizzati per verificare se un nuovo piano riesce a raggiungere il goal allo stato attuale. Si fa degli altri agenti che collaborano al raggiungimento di un goal comune. Al momento non viene ancora gestita questa modalità di planning, ma si procede alla ripianificazione del solo agente che ha individuato l'impossibilità di raggiungere in tempo il goal, considerando che gli altri agenti continuino a mantenere il piano da loro prefissato, in totale autonomia.

Il ciclo di sviluppo e verifica dei sistemi MAS in REMM è mostrato in Figura 2.

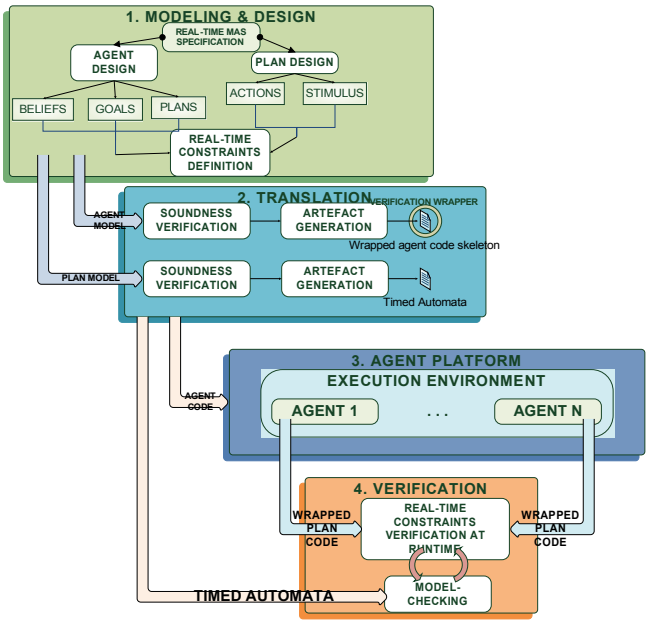

Figura 1: Ciclo di Sviluppo

Il ciclo di sviluppo consta di quattro fasi:

1. **Modeling & Design:** la prima fase consiste nel modellare il sistema, descrivendo dal punto di vista strutturale gli agenti che lo costituiscono ed i piani che essi implementano. Pertanto, una volta stilato il

documento di specifica del sistema multi-agente real-time, si passa alla modellazione vera e propria, che si sviluppa in due passi: L'agent Design, dove viene progettato l'agente e il Plan Design, dove vengono progettati i suoi comportamenti.

2. **Translation:** i modelli prodotti nella fase precedente sono opportunamente processati al fine di generare nuovi artefatti. Per essi viene effettuata una verifica di correttezza, in modo da stabilire la fattibilità del processo di traduzione. Il processo utilizza tecniche di model checking[7] applicate a modelli timed automata [8,9]
3. **Agent Implementation and Execution:** In questa fase si procede alla scrittura del codice che implementa l'agente, che viene integrato con dei sistemi di verifica a run-time dipendenti dal modello che li ha generate.
4. **Verification:** il processo di verifica consiste nel determinare la fattibilità dell'esecuzione dei piani di ogni agente nel rispetto dei vincoli temporali imposti.

3.Il Framework REMM

I Componenti principali del Framework REMM sono mostrati in Figura 3.

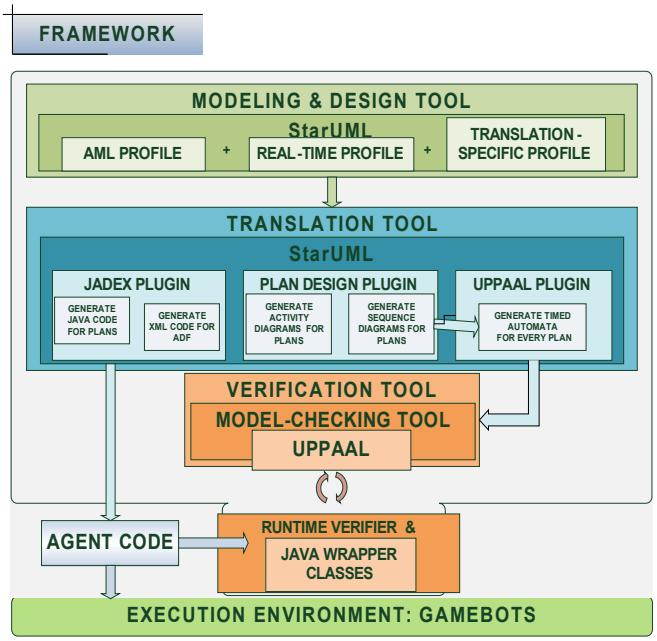

Figura 2: Componenti di REMM

Il **Modeling & Design Tool** è stato realizzato utilizzando il tool StarUML[12], su cui è stato sviluppato un plugin per la modellazione utilizzando un dialetto UML (l'Agent Modeling Language, AML), modificato per supportare esplicitamente i vincoli real-time nella definizione dell'agente. Oltre al profile AML, quindi, sono stati implementate due estensioni di questo linguaggio, definendone formalmente la semantica: il real-time profile, arricchisce una parte di AML (quella relativa alla modellazione di agenti BDI) con informazioni relative al tempo; il platform-specific profile, in cui sono definiti nuovi stereotipi atti a modellare i diversi aspetti di una piattaforma di riferimento.

In questo modo e' possiblie definire dei diagrammi simili ai class diagrams di UML per la definizione statica di believes, desires and intensions dell'agente, mentre la definizione dimanica viene demandata a diagrammi simili agli activity e ai sequence.

Il Translation Tool è strumento di traduzione e generazione di codice è integrato nel tool di modellazione StarUML sottoforma di plugin, ed è scritto interamente in Jscript. In particolare il Framework consta di tre plugin adibiti al processo di traduzione e generazione: il Jadex Plugin è preposto alla verifica di correttezza e alla traduzione degli agenti BDI modellati mediante l'Agent Diagram, e relativi alla piattaforma Jadex; Il Plan Design Plugin è preposto alla generazione dei diagrammi di attività e di sequenza, previa lettura del modello definito nell'Agent Diagram; l' UPPAALPlugin è preposto alla generazione di Timed e processabili dal model-checker UPPAAL[10] ai fini della verifica del modello AML-Real-Time.

Il Verification Tool e Run Time Verifier: per implementare il processo di verifica si fa uso di due strumenti che interagiscono tra di loro: uno schedulatore/verificatore dinamico ed il model-checker Uppaal. In questo componente, il Runtime Verifier & Scheduler esegue la verifica dei modelli AML-Real-Time in fase di progetto, e a run time, utilizzando un insieme di classi wrapper scritte in Java, le quali mettono a disposizione i metodi start e stop per avviare e chiudere il processo di verifica a run-time. Le proprietà da verificare sono proprietà di raggiungibilità descritte in logica temporale TCTL. La necessità di una verifica anche a run-time è dovuta al fatto che il comportamento reale di un sistema, potrebbe discostarsi da quello del modello. Il verificatore controlla che i vincoli temporali specificati in fase di progetto siano ancora validi anche quando le proprietà temporali dei sistemi in esecuzione si discostano da quello del modello. UPPAAL viene utilizzato ancora per la verifica delle proprietà temporali. Lo scheduler e il verificatore a run-time vengono implementati generando un codice per la piattaforma JADEX[6,11]. Di seguito viene mostrato un esempio di utilizzo di REMM per lo sviluppo di BOTS intelligenti per il videogame Unreal-Tournament

4.Caso di Studio

Come caso di Studio si è scelto lo sviluppo di due GAMEBOTS[13] per Unreal-Tournament. La piattaforma GAMEBOTS supporta nativamente JAVA per lo sviluppo dei Bots, ed è stato facile interfacciarlo con il framework REMM.

I due GAMEBOTS da sviluppare, sono rispettivamente un agente che cerca il giocatore per ucciderlo, e un agente che cerca l'agente attaccante per offrirgli un supporto (virtuale). Le azioni dell'agente attaccante sono sottoposte a vincoli real-time sia nella fase di ricerca dell'avversario, sia nella fase in cui cerca di ucciderlo senza subire grossi danni.

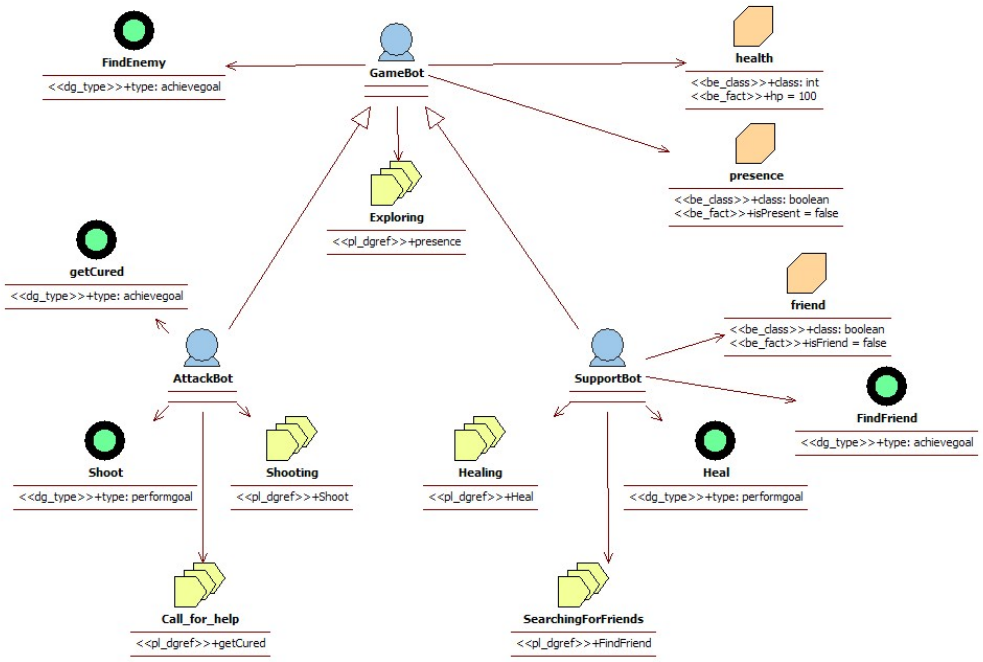

Figura 4: Diagramma Statico di Bots

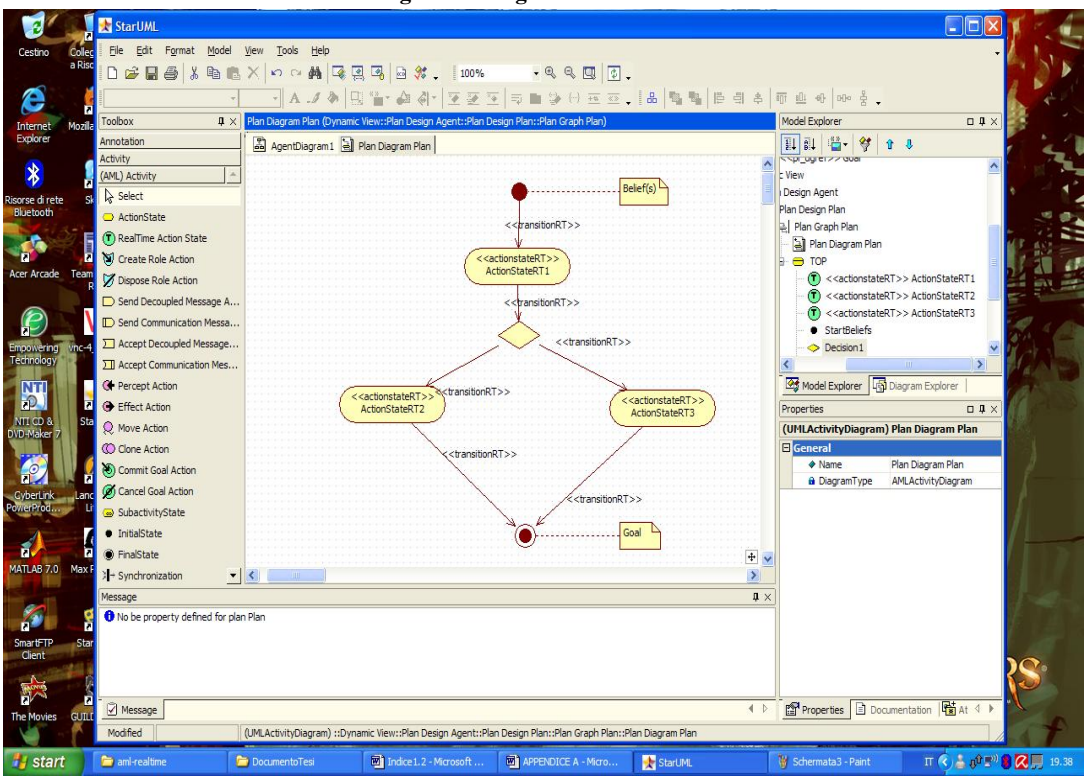

Figura 5: Un Diagramma Dinamico

La prima fase del ciclo di sviluppo dei bots prevede la definizione del loro modello in StarUML. In Figura 4 e 5, sono ad esempio riportati un diagramma statico ed uno dinamico dei bots. In Figura 4, gli omini stilizzati rappresentano le classi agente, I cerchi concentrici i loro goals, le figure pentagonali gialle i loro piani e le figure esagonali arancioni alcuni dei loro beliefs. Il diagramma dinamico in Figura 5 è invece un activity diagram esteso con proprietà real-time.

Tutti i diagrammi definite in questa fase, vengono elaborati per produrre dei modelli basati su Timed Automata che verificano le proprietà di raggiungibilità dei goals dati i vincoli real-time. Nel caso questi vincoli vengano verificati, si procede alla generazione

degli stub software JADEX con integrati il verificatore a run-time e lo schedulatore, cosi' come mostrato in Figura 6.

```
package StarUMLtoJADEX;

import jadex.runtime.*;
import java.util.*;
import edu.wrapper.modelchecking.*;
import edu.wrapper.action.*;

public class PlanPlan extends Plan {

    Map timeMap = getTimeoutMap();

    //This wrapper executes runtime model checking...
    modelCheckWrapper wrap;
    long startTime;
    long goalDeadline;

    public void body() {

        //Application code goes here...

        //Be careful: you must define startTime and goalDeadline!

        wrap = new modelCheckWrapper(startTime, goalDeadline);

        try {

            wrap.start();
            ActionStateRT1();
            wrap.stop(timeMap.get("ActionStateRT1"));

            wrap.start();
            ActionStateRT2();
            wrap.stop(timeMap.get("ActionStateRT2"));

            wrap.start();
            ActionStateRT3();
            wrap.stop(timeMap.get("ActionStateRT3"));

        } catch(PlanFailureException pfe) { /*Here you can drop goal, for example...*/ }

    }
```

Figura 3: Stub Scheduler e Verificatore

Il Bot viene poi completato con l'inserimento delle librerie di GAMEBOTS. I Bot completi vengono integrati all'interno di Unreal-Tournament e rilasciano periodicamente informazioni sull'esecuzione del verificatore e dello scheduler, cosi' come mostrato in Figura 7.

Figura 4: il BOT

5.Conclusioni

Il Framework REMM è stato realizzato avendo in mente i seguenti obiettivi: fornire uno strumento CASE per la modellazione e progettazione di sistemi multi-agente operanti in tempo reale; fornire uno strumento di traduzione del modello di cui sopra per una piattaforma ad agenti di riferimento; fornire uno strumento di verifica dei vincoli real-time a tempo di progetto ed esecuzione. Come sviluppi futuri si vuole introdurre delle politiche di replanning basate sui controesempi del model checker in modo da scegliere, in caso di problemi temporali, il miglior piano da intraprendere per raggiungere comunque il goal iniziale, o per scegliere definitivamente di cambiare l'obiettivo per intraprendere, ad esempio, azioni di repair o per portarsi in uno stato sicuro.

6.Bibliografia

1.M. d'Inverno, M. Luck, M. Georgeff, D. Kinny, M. Wooldridge (2004), The dMARS Architechure: A Specification of the Distributed Multi-Agent Reasoning System, J. of Autonomous Agents and Multi-Agent Systems, 9(1-2):5-53.

2.J. Himoff, P. Skobelev, M. Wooldridge(2005), MAGENTA Technology: Multi-Agent Systems for Industrial Logistics, Proceedings of the AAMAS 2005 Industry Track, .

3.Krishna Kavi, Mohamed Aborizka and David Kung (2002), A framework for designing, modeling and analyzing agent based software systems, in Proc. of 5th International Conference on Algorithms & Architectures for Parallel Processing, October 23-25, 2002, Beijing, China

4.V. Silva and C. Lucena (2004), From a Conceptual Framework for Agents and Objects to a Multi-Agent System Modeling Language, Journal of Autonomous Agents and Multi-Agent Systems 9(1-2), Kluwer Academic Publishers, p. 145-189.

5.Lichen Zhang (2006), "Development Method for Multi-Agent Real-Time Systems", International Journal of Information Technology, Vol. 12 No. 5 2006.

6.A. Pokahr, L. Braubach, W. Lamersdorf (2005), "Jadex: A BDI Reasoning Engine", Chapter ofMulti-Agent Programming, Kluwer Book, Editors: R. Bordini, M. Dastani, J. Dix and A. Seghrouchni.

7.M. Wooldridge, M.-P. Huget, M. Fisher, S. Parsons (2006). Model Checking Multi-Agent Systems: The MABLE Language and Its Applications, International J. on Artificial Intelligence Tools, 15(2):195-225.

8.M. Wooldridge (2003), An Automata-theoretic approach to Multiagent planning, Proceedings of the First European Workshop on Multiagent Systems (EUMAS 2003).

9.R. H. Bordini, M. Fisher, W. Visser, & M. Wooldridge (2004), Verifiable multi-agent programs, in M. Dastani, J. Dix, & A. El Fallah-Seghrouchni, (Eds.) Programming multi-agent systems, proceedings of the first international workshop (ProMAS-03), 15 July, 2003, Melbourne, Australia, number 3067 in Lecture notes in artificial intelligence, (pp. 72-89) Berlin, Springer-Verlag

10.Gerd Behrmann, Alexandre David, Kim G. Larsen (2004), A Tutorial on Uppaal, Department of Computer Science, Aalborg University, Denmark.

11.F. Bellifemine, G. Caire, A. Poggi, G. Rimassa, JADE - A white paper, in EXP in search of innovation - Special Issue on JADE TILAB Journal, 2003.

12.Wong S. StarUML Tutorial [Connexions Web site]. September 10, 2007. Available at: http : //cnx.org/content/m15092/1.1/.

13.Kaminka, G. A.; Veloso, M.; Schaffer, S.; Sollitto, C.; Adobbati, R.; Marshal,Andrew N.; Scholer, Andrew, S.; and Tejada, S. 2002. GameBots: the everchallenging multi-agent research test-bed, In Communications of the ACM,January issue.

Questo Lavoro è stato sviluppato nell'ambito del progetto "Magda una piattaforma ad agenti per il GRID computing" L.R.5/02 della Regione Campania, 2005