

M&S using MAS

Andrea Tosi

12 febbraio 2010

Indice

1	Introduzione	3
2	Modellazione Agent-Based	3
3	Framework di Zeigler	5
3.1	Riflessioni	6
3.1.1	Accomodation relation	7
3.2	La replicazione per simulatori MAS	7
4	Complessità dei modelli	9
4.1	Ragioni soggettive	9
4.2	Ragioni tecniche	9
5	Linee guida per la modellazione	9
5.1	Model Composition	10
5.1.1	BOM-based Model Composition	10
6	Linee guida per la progettazione	11
7	Conclusioni	12

1 Introduzione

La simulazione è oggi sempre più utilizzata sia come appoggio alla sperimentazione fisica sia come strumento nella fase di testing di sistemi software complessi.

Per questo l'M&S (Modeling and Simulation) viene ormai considerato una scienza a tutti gli effetti.

La presente relazione analizzerà le interconnessioni tra i MAS ed il mondo del M&S. In tale settore, infatti, i MAS possono essere impiegati sia come paradigma per la modellazione del sistema reale, sia come paradigma per l'implementazione del simulatore.

Nella prima parte si riporterà un framework concettuale per l'analisi del processo di modellazione e simulazione e verranno analizzati pregi e difetti dell'utilizzo dei MAS in tale contesto.

Nella seconda parte si presenteranno due possibili approcci alla complessità realizzativa di un simulatore agent-based.

2 Modellazione Agent-Based

Shannon (1975) definiva la simulazione come:

“The process of designing a model of a real system and conducting experimentes with this model for the purpose either of understanding the behavior of the system and/or of evaluating various strategies (withing the limits imposed by a criterion or a set of criteria) for the operation of the system.”

Volendo quindi definire il processo di simulazione, le fasi principali sarebbero:

- comprendere il sistema reale/virtuale e definirne un modello;
- costruire un simulatore, effettuare esperimenti ed analizzare i risultati.

Soffermiamoci sulla prima fase.

I primi simulatori utilizzavano sistemi di equazioni differenziali (Equation-Based Simulation) per modellare i sistemi reali; successivamente, avendo intuito la causalità intrinseca dei fenomeni e avendo a disposizione i primi calcolatori, furono formulate e introdotte nel modello equazioni stocastiche che, pur avendone migliorato la qualità, non resero il modello in grado di cogliere alcuni aspetti fondamentali del fenomeno in esame, poichè [Feber, 1999]:

- è possibile analizzare unicamente la prospettiva globale del sistema;
- le equazioni non prendono in considerazione la complessità delle dinamiche delle interazioni di micro-livello;
- è difficile modellare aspetti qualitativi del sistema.

Il primo settore in cui si è cercato di porre rimedio a tali mancanze è stato quello della simulazione di sistemi economici [Orcutt, 1957]. Si era infatti notato che alcuni fenomeni di macroeconomia non potevano essere spiegati senza analizzare le dinamiche tra micro-entità quali i cittadini e le imprese.

Nasce così la microsimulazione che riconosce la presenza di due livelli: un *micro-livello* composto da entità, il cui comportamento è determinato da regole deterministiche o statistiche, e un *macrolivello* in cui si definiscono le regole e le proprietà

globali del sistema. I due livelli sono strettamente collegati in quanto le dinamiche del secondo possono essere considerate come il *comportamento emergente* del primo.

La nascita della microsimulazione ha posto le basi per l'avvento della modellazione agent-based (ABM) in cui non ci si limita a modellare unicamente i comportamenti delle entità, ma si esplicitano le interazioni e le azioni che le entità compiono sull'ambiente di simulazione. Il comportamento globale del sistema è dato dall'insieme di queste micro-interazioni.

Ogni agente può essere caratterizzato da quattro aspetti:

autonomia: le azioni compiute dall'agente sono mirate ad ottenere un certo obiettivo (goal). L'agente in maniera autonoma regola il suo comportamento, utilizza risorse e comunica con altri agenti;

comportamento sociale: ogni agente è in grado di interagire con altri agenti. Il comportamento emergente è la nascita di una o più società;

situadness: ogni agente è situato in un ambiente ed interagisce con esso attraverso azioni e percezioni. L'ambiente è una sorta di colla che permette l'interazione.

Modellare il sistema come insieme di entità autonome, interagenti tra loro e situate in un certo ambiente, permette di creare dei micro-cosmi artificiali che riproducono, in piccolo, il comportamento di un sistema complesso.

Il MAS (Multi-Agent System), che così si crea, è estremamente flessibile in quanto consente di: modificare o analizzare il comportamento di ogni singolo individuo, analizzare il sistema globalmente o focalizzarsi sulle interazioni agente-agente/agente-ambiente.

Il collegamento tra individuale e collettivo è il punto cardine che permette di affrontare la complessità di alcuni fenomeni reali.

Tuttavia, l'utilizzo di un MAS obbliga il modellista a compiere alcune scelte prima di procedere con la progettazione.

• ARCHITETTURA AGENTE

- reattiva: l'agente non ha una rappresentazione esplicita né dell'ambiente né degli altri agenti. Il suo comportamento è totalmente descritto da una lista di reazioni in base a stimoli;
- cognitiva: l'agente ha un'esplicita rappresentazione dell'ambiente e degli altri agenti. Il suo comportamento si fonda su cicli di *ragionamento* relativi a una base di conoscenza simbolica. In conseguenza al ciclo di reasoning l'agente determina l'attuazione di piani che si risolvono in azioni sull'ambiente (BDI);
- ibrida: l'architettura cognitiva e reattiva sono viste come complementari. Problema fondamentale è determinare il giusto bilanciamento tra le due attività.

- **COMPORTAMENTO AGENTE:** occorre fissare quella che viene chiamata *deliberation function*, ossia la funzione che permette di collegare le percezioni dell'agente alle azioni da esso compiute sull'ambiente. Il comportamento ciclico dell'agente è, in questo caso, composto da tre fasi:

1. PERCEZIONE

2. DELIBERAZIONE

3. AZIONE

- **AMBIENTE:** è parte fondamentale del sistema in quanto conferisce la situadness agli agenti e rappresenta l'ambiente di simulazione. Occorre fissarne le proprietà fondamentali:
 - Accessibile vs Non accessibile: il suo stato è/non è completamente accessibile agli agenti;
 - Deterministico vs Non deterministico: lo stato futuro dell'ambiente è/non è completamente determinato dallo stato presente e dell'insieme di azioni compiute dagli agenti;
 - Episodico vs Non episodico: il risultato di un'azione effettuata sull'ambiente dipende/non dipende da quelle passate;
 - Statico vs Dinamico: i cambiamenti dell'ambiente dipendono solo/non dipendono dalle azioni degli agenti;
 - Discreto vs Continuo: il numero di possibili percezioni di un agente è finito/infinito;
- **MODELLO DI COMUNICAZIONE:** è necessario fissare le modalità con le quali gli agenti comunicano tra di loro: message passing, segnali, etc. . .
- **MODELLARE IL TEMPO:** esistono tre modi principali:
 - tempo continuo: esiste una funzione in grado di computare lo stato del sistema in ogni istante di tempo (sistemi DES);
 - tempo discreto: il tempo evolve ad intervalli discreti;
 - tempo discreto event-based: il tempo evolve in maniera discreta da un evento all'altro considerando una linea temporale continua.

Il modellista dovrà fissare come viene considerato il tempo all'interno dell'agente, all'interno dell'ambiente e nell'interazione agente-ambiente (azioni).

- **ACCOPPIAMENTO AGENTE-AMBIENTE:** consiste nel determinare la funzione di *evoluzione* dell'ambiente in base allo stato di esso, alle azioni compiute dagli agenti e ai vincoli temporali.

3 Framework di Zeigler

Zeigler nel 2000 propone un framework per la modellazione e la simulazione (M&S) che ha come scopo quello di definire le fasi fondamentali per la realizzazione di un *sistema di simulazione*.

Si introducono quattro entità:

Sistema Sorgente: il sistema in esame può essere sia reale che virtuale, in generale rappresenta il sistema target della simulazione.

Experimental Frame: formalmente rappresenta l'insieme di condizioni in cui il sistema viene osservato e sperimentato. Esplicità gli obiettivi della simulazione.

Modello: termine a questo punto ambiguo. Rappresenta formalmente l'insieme di tutti i possibili stati e transizioni del sistema, cioè la *funzione di evoluzione* che, nel caso si utilizzi una modellazione multi-agente, viene fissata nella fase di scelta dell'accoppiamento agente-ambiente.

Simulatore: consiste nel *valutatore* del modello (funzione di evoluzione).

Successivamente Zeigler introduce varie *relazioni* tra queste entità.

In primo luogo viene definita la relazione di *validità* che lega il modello all'experimental frame e quindi al sistema sorgente. Si considerano tre tipi di validità (quelle successive implicano quelle precedenti):

Replicativa: il comportamento del modello corrisponde con una certa approssimazione a quello del sistema in esame.

Predittiva: il modello è in grado di predire comportamenti non osservati nel sistema sorgente. Quindi, le transizioni di stato del modello avvengono al fine di riprodurre i comportamenti del sistema sorgente ad ogni possibile input.

Strutturale: il modello mima gli stessi cambiamenti di stato del sistema sorgente.

Inoltre, viene definita la relazione di *correttezza del simulatore* che intercorre tra il simulatore e il modello in cui si analizza la capacità del simulatore di generare correttamente il comportamento del modello. In altre parole, essendo il simulatore un software, occorre che il modello descriva le proprie transizioni interne in maniera non ambigua, computazionalmente comprensibile ed uniforme.

3.1 Riflessioni

Nella sua descrizione Zeigler specifica che il sistema sorgente può essere un sistema reale o virtuale. Allora, *come distinguere i due tipi?*

Analizzando le relazioni di validità si deduce che, nel caso in cui il sistema sorgente sia virtuale, la validità strutturale non può essere determinata in quanto il sistema è il modello stesso; mentre la validità replicativa, per lo stesso motivo, è sempre valida. Inoltre si possono avere casi in cui il sistema reale è modellato in maniera talmente astratta da essere considerato puramente teorico (vedi simulazione di comportamenti sociali).

È quindi utile introdurre un criterio preciso per classificare il sistema sorgente: il *behavior database*. Esso consiste nell'insieme dei comportamenti del sistema sorgente.

Se il behavior database può essere fissato prima o dopo la simulazione, il sistema sorgente può essere considerato reale; in caso contrario virtuale. Potendo individuare tutti i comportamenti del sistema sorgente è, infatti, possibile applicare le relazioni di validità al modello. Nel caso di un Agent-Based Model dobbiamo considerare l'ambiente come parte distinta del modello e del sistema sorgente. Possiamo quindi avere quattro possibili configurazioni:

- agente virtuale in un ambiente reale
- agente reale in un ambiente virtuale
- agente virtuale in un ambiente virtuale
- agente reale in un ambiente reale

Ogni variante porta con sé problematiche specifiche di validazione che il modellista dovrà affrontare.

Considerando ancora una volta le relazioni di validità introdotte da Zeigler un'ulteriore domanda che sorge è la seguente:

in che modo è possibile determinare la validità replicativa di un modello?

Per rispondere è necessario dividere i modelli in due categorie:

1. modelli di sistemi computazionali reali;
2. modelli di sistemi viventi reali;

Nel primo caso la risposta è ovvia: il sistema in esame è composto da strutture note; il comportamento del modello può essere quindi paragonato facilmente al comportamento del sistema sorgente.

Nel secondo caso le entità del sistema sorgente sono black boxes di cui conosciamo più o meno chiaramente il contenuto: paragonare il comportamento del modello a quello del sistema reale diventa a questo punto molto complesso.

Nella maggior parte dei casi la validazione è soggettiva, questo porta il modello scelto a non essere uniformemente accettato dalla comunità scientifica.

Nel caso di MABS (Multi-Agent Based Simulation) il problema della validità replicativa è estremamente sentito. Una delle soluzioni più interessanti è stata quella proposta da Guyot e Honiden nel 2006: *l'Agent-Based Participatory Simulation*. L'idea è unire il processo di design di un simulatore ad una sorta di gioco RPG (Role Playing Game), quindi unire la fase di validazione a quella di progettazione del modello in un processo interattivo con la comunità. Le persone che partecipano al "gioco" comandano attivamente un agente interno alla simulazione. I risultati dell'esecuzione vengono memorizzati e analizzati al fine di modellare al meglio il sistema artificiale.

3.1.1 Accomodation relation

Dato un paradigma di modellazione è interessante analizzare quella che Zeigler definiva come la *accomodation relation* che sussiste tra l'experimental frame ed il modello. Essa valuta la corrispondenza tra il paradigma utilizzato per la modellazione e il fenomeno da simulare.

Un tipo di simulazione agent-based implicitamente obbliga ad utilizzare un MAS come paradigma di modellazione. Questo porta il modellista a dover rispettare in primo luogo le regole del paradigma e solo successivamente gli obiettivi della simulazione. Rimane da capire la reale potenza espressiva del paradigma ad agenti.

David nel 2002 fa notare che esiste un gap tra ABM ed experimental frames, dovuto alla mancanza di un legame esplicito tra il micro e il macro livello nella simulazione.

In questa prospettiva il problema non è quindi modellare l'autonomia, o aspetti legati al singolo, ma introdurre direttamente in fase di modellazione comportamenti emergenti come: stigmergia, self-organization, etc. . .

3.2 La replicazione per simulatori MAS

Viene definita *replicazione* la capacità di un simulatore di eseguire modelli differenti.

È possibile individuare tre tipi di simulatori basati su MAS:

One Shot: limitati ad una sola classe di modelli.

Domain Related: accettano più modelli di uno stesso dominio.

Generic: progettati per essere implementati con una particolare tecnologia (es. Jason) indipendenti dal dominio di applicazione.

I simulatori più diffusi oggi sono quelli del primo tipo.

Nella maggioranza dei casi occorre realizzare un nuovo simulatore per ogni nuovo modello.

Uno dei motivi potrebbe essere la necessità di adattare il MAS ad ogni modello (questo comporta la re-implementazione del simulatore); oppure considerare il gap tra modello e paradigma multi agente che rende difficilmente comprensibile i framework a disposizione. Non sorprende che ogni ricercatore preferisca realizzare tanti simulatori one shot piuttosto che impiegare tempo ad adattare un framework già esistente.

Purtroppo i simulatori one shot non rispettano la relazione di replicazione. Questo porta al problema dell'*engineering divergence phenomenon* [Michel et al., 2004] es. l'impossibilità di implementare simulatori differenti considerando lo stesso modello.

Possiamo quindi individuare un secondo gap che interessa l'utilizzo di MAS: quello tra il paradigma ad agenti e le tecnologie a disposizione.

Uno degli aspetti fondamentali completamente trascurati nei simulatori one shot è quello dello *scheduling*. Per assicurare la replicazione, oltre ad esplicitare in maniera formale il modello utilizzato, è essenziale esplicitare anche le modalità con cui il modello viene poi eseguito dal simulatore. Con tali informazioni in possesso sarebbe possibile replicare i risultati ottenuti su molti simulatori facilitando così la condivisione della conoscenza nella comunità scientifica.

Possiamo quindi individuare due problemi che rendono difficile la replicazione dei simulatori multi-agente:

1. mancanza di un formalismo uniforme per la descrizione dei modelli;
2. mancanza di una metodologia uniforme per la progettazione di simulatori basati su MAS.

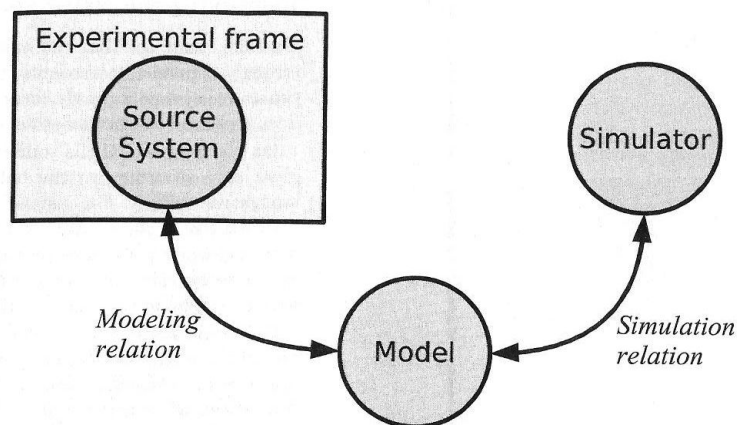


Figura 1: Framework di Zeigler

4 Complessità dei modelli

Non esiste una singola ed accettata definizione di complessità applicabile ad un modello. Alcune definizioni considerano “complesso” un modello composto da molti componenti, altre prendono in considerazione l’aspetto cognitivo ossia la difficoltà di comprendere il sistema reale dato il modello. Ward nel 1989 comprendeva entrambi gli aspetti definendo la complessità come la capacità di un modello di essere *trasparente* (aspetto cognitivo) e *strutturalmente semplice*.

Ci sono stati tentativi di definire formalmente una scala valutativa della complessità di un modello [Wallace, 1987] ma non sono riconosciuti validi.

Negli ultimi anni si sta assistendo alla crescita della complessità dei modelli proposti. Le ragioni possono molteplici, tutte legate da una problematica comune: *la tendenza è quella di soffermarsi sullo studio della struttura del sistema da modellare piuttosto che sugli obiettivi della simulazione*.

4.1 Ragioni soggettive

Fattore “show-off”: il modellista è portato a realizzare un modello complesso al fine di valorizzare il suo lavoro davanti ad un manager o una commissione.

Sindrome “includi tutto”: i modellisti inesperti durante la realizzazione del modello tendono ad integrare aspetti inutili per non rischiare di tralasciare elementi del sistema poco chiari o che hanno dimenticato. Molto spesso questo errore si verifica quando la scadenza del lavoro è ravvicinata.

Fattore di “possibilità”: c’è la tendenza ad includere nel modello aspetti non necessari poiché si è convinti che la potenza di calcolo della CPU su cui il modello verrà applicato, lo consentirà (essendo essa in continua crescita).

4.2 Ragioni tecniche

Scarsa conoscenza del sistema da modellare: in questo caso, per non commettere errori, la tendenza è di accorpare nel modello il maggior numero di elementi possibili.

Incompetenza a modellare correttamente il problema: il modellista tende a costruire un modello che include praticamente ogni aspetto del mondo, anche quelli che non si desidera analizzare.

Mancanza di paradigmi computazionali adatti: errori di programmazione, oltre ai problemi già discussi precedentemente.

Obiettivo della simulazione non chiaro: è un’idea condivisa tra alcuni autori: *specificare in maniera incompleta gli obiettivi della simulazione porta all’ingigantimento del modello e dello scope di simulazione (experimental frame)*.

5 Linee guida per la modellazione

Si presentano alcune regole per una buona modellazione:

Keep it simple: se il modello nasce semplice ne guadagna l'intero processo di studio del fenomeno.

Add complexity later: se si hanno dubbi, non aggiungere nuovi elementi al modello. Si deve assumere che questi non influenzino il risultato. *Be parsimonious, start small and add.*

Using Hierarchy: in tal modo si riduce il livello di dettaglio. Si ottiene una vera semplificazione solo se è possibile aggregare tra loro parti del modello.

Reduce the scope: (scope=sperimental frame) riducendo il sistema da simulare la complessità del modello diminuisce.

5.1 Model Composition

Il processo di modellazione di un sistema reale può risultare lungo e complesso. Ogni modello può essere considerato come la composizione di elementi indipendenti: tenendo conto della riusabilità di tali componenti, è possibile pensare ad un'ingegnerizzazione del modello che ha come scopo quello di ridurre i tempi e i costi associati alla fase di modellazione.

La *model composition* è un approccio che permette di comporre un modello con un insieme di componenti già validati in precedenti simulazioni, a disposizione del modellista.

Il comportamento del modello, così costruito, dovrà essere la composizione dei comportamenti dei singoli componenti. Tale proprietà viene detta *composability* e si divide in:

- composability sintattica che riguarda l'aspetto implementativo della composizione delle parti del modello;
- composability semantica che prende in esame il significato associato alle singole componenti e come questo sia in relazione al modello risultante.

La prima può essere ottenuta sfruttando la software engineering, la seconda è materia di ricerca.

5.1.1 BOM-based Model Composition

Un requisito dell'analisi della replicazione dei simulatori MAS è la necessità di un linguaggio formale per la rappresentazione di un modello: il BOM nasce proprio per questo.

Il Base Object Model è uno standard SISO (Simulation Interoperability Standards Organization). Si basa su documenti XML che descrivono i componenti di un modello per la simulazione.

Il funzionamento del modello è inteso come insieme di pattern di simulazione ovvero flussi di eventi tra componenti.

Un BOM si divide in quattro parti:

Model Identification: contiene i metadata del modello.

Conceptual Model: contiene i pattern di simulazione, ognuno dei quali espresso come:

- una descrizione
- una macchina a stati
- la lista di entità di cui è composto
- il flusso di eventi tra le entità

Model Mapping: determina le corrispondenze tra il conceptual model e l'HLA OMT (High Level Architecture Object Model Template).

HLA Object Model: architettura di più basso livello utile all'implementazione del simulatore.

Il BOM è uno standard estremamente articolato, per questo se ne riportano unicamente le caratteristiche principali e la struttura.

È intuibile che utilizzando tale formalizzazione si analizzano il funzionamento di un modello e le sue proprietà, a monte della realizzazione completa del simulatore: questo riduce i tempi di sviluppo del sistema. Esplicitando l'architettura HLA OMT-Based del simulatore, si associano al modello quelle informazioni di *scheduling* che facilitano la replicazione del simulatore.

6 Linee guida per la progettazione

In questa sezione viene riportata una possibile metodologia per la progettazione di un simulatore multi-agente.

Come già visto, è importante separare l'infrastruttura del simulatore dal modello da simulare è quindi utile applicare la relazione di composizione, oltre che al modello, anche all'architettura del simulatore.

Consideriamo un simulatore multi-agente come composto da componenti software ognuno dei quali:

- è un'entità separata del deployment;
- può essere realizzato da terze parti;
- ha un'interfaccia uniforme;
- ha un funzionamento che dipende da una particolare piattaforma.

È d'obbligo introdurre assunzioni che riguardano l'architettura del MAS che si va a realizzare:

Ambiente: rete di “relazioni sociali” che intercorrono tra gli agenti del sistema.

Comunicazione: ogni agente comunica con quelli a cui è collegato tramite *stimoli*.

Si individuano inoltre i seguenti *artefatti*:

Componenti Software per la simulazione che si dividono in:

- Agenti.
- Proprietà: le misure che possono essere calcolate su un agente.

Ogni componente durante la simulazione verrà “istanziato” e sarà in grado di comunicare con altri.

Scenari: quanti e quali componenti dovranno essere “istanziati” e come essi verranno collegati.

Esperimenti: rappresentano, insieme con lo scenario, la descrizione della funzione di scheduling del simulatore; descrivono quali azioni eseguire su un particolare scenario.

Il MAS è costruito collegando più *componenti* tra loro in un particolare *scenario* relativamente ad un particolare *esperimento* il tutto tradotto in un particolare formalismo.

La metodologia proposta presenta molti pregi.

In primo luogo permette di esplicitare in maniera formale la funzione di scheduling. Inoltre separa concretamente le entità in gioco dalle loro relazioni. Per esempio si potranno utilizzare nella stessa simulazione agenti realizzati con architetture diverse.

Tuttavia non viene considerato l'ambiente come entità indipendente di un MAS: questo può essere considerata una grave mancanza in quanto si perde il concetto di situadness degli agenti ed il sistema deficiata di capacità informativa.

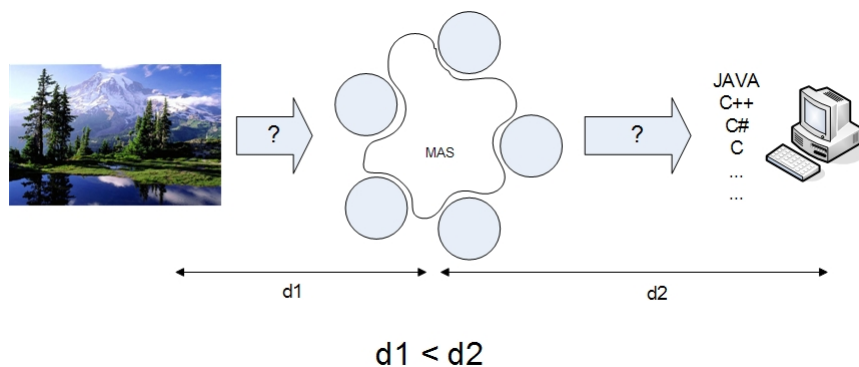
Secondo chi scrive per far fronte di tale deficit, si è introdotto un componente delle proprietà (insieme di attributi osservabili di un agente); contemporaneamente, però, vengono definiti gli agenti come black boxes, senza proprietà osservabili se non quelle risultanti dalle interazioni con altri agenti. A prima vista, associare proprietà osservabili ad un agente è contro il paradigma; d'altronde è utile ai fini della simulazione. A fronte di una maggiore complessità progettuale, modellare l'ambiente come entità di prima classe risolverebbe il problema: da un lato fornirebbe un maggiore apporto informativo alla simulazione, dall'altro rimarrebbe congruente con il paradigma.

7 Conclusioni

Nelle sezioni precedenti si sono evidenziati e analizzati due grandi gap che impediscono l'utilizzo intensivo dei MAS per la simulazione:

1. il modellista, scegliendo il paradigma ad agenti, è costretto a fissare alcune assunzioni sul modello che allontanano l'attenzione dai veri obiettivi della simulazione: il gap si pone tra la potenza del paradigma ad agenti e la necessità espressiva del modellista. Tale divario è analizzato dall'*accomodation relation* nel framework di Zeigler;
2. una volta modellato il sistema come MAS, l'implementazione del simulatore è un processo che risente del gap tra paradigma ad agenti e tecnologie disponibili.

Tali gap influenzano sia la validità del modello sia la replicazione del simulatore.



Nelle sezioni precedenti si sono citate tecniche e approcci per diminuire questi divari ed in generale facilitare la replicazione per un simulatore MAS-based.

È stato citato BOM come formalismo per una rappresentazione uniforme di un modello per la simulazione. Successivamente è stata proposta una metodologia progettuale che permette un collegamento più agile tra la fase di modellazione del sistema e di progetto del simulatore.

Consultando la letteratura è possibile capire che l'approccio principale alla complessità della modellazione del mondo reale è l'introduzione di nuovi paradigmi sempre più potenti al fine di aumentare la potenza espressiva del modello. Purtroppo le tecnologie a disposizione usualmente non evolvono rapidamente come i paradigmi e, quasi sempre, si cerca di adattare sistemi e linguaggi già esistenti al fine di riprodurre il comportamento desiderato.

Per questo, sempre più frequentemente, esiste la necessità di introdurre nuove strutture computazionali che meglio rispecchino il vero comportamento del modello. In altre parole, piuttosto che adattare vecchi paradigmi o linguaggi al fine di modellare comportamenti reali, si dovrebbero utilizzare comportamenti reali per modellare nuove strutture computazionali.