Modelling and Analysis of Agent-Based Electronic Marketplaces

Giancarlo Fortino, Alfredo Garro, and Wilma Russo

Abstract—*In this paper, an approach for modelling and analysing e-Marketplaces based on agents is proposed. The approach is founded on a Statecharts-based specification language and on a Java-based, discrete event simulation framework. The former allows for the modelling of the behaviour of the main agents (stationary and mobile) of an e-Marketplace along with their interaction protocols. The latter supports the execution through simulation of agent-based e-Marketplace models. The approach is exemplified by defining and simulating a consumer-driven e-Marketplace model which offers mobile agent-based services for searching and buying goods. The simulation phase enabled validation of the e-marketplace model and evaluation of the performances of different kinds of mobile consumer agents.*

Index Terms— ***Mobile Agents, Statecharts, Agent-based e-Marketplace, Event-driven Simulation, Performance Evaluation, Java***

1. INTRODUCTION

ELECTRONIC Marketplaces (e-Marketplaces) are e-commerce environments which offer new channels and business models for buyers and sellers to effectively and efficiently trade goods and services over the Internet [12]. To support intelligent and automated e-commerce services, new enabling infrastructures are needed.

Manuscript received September 5, 2004.

Giancarlo Fortino is Assistant Professor of Computer Science Dipartimento di Elettronica, Informatica e Sistemistica (DEIS) , Italy. (e-mail: g.fortino@unical.it)

These infrastructures can be effectively developed using the emerging Agent technology and paradigm [10] along with XML-based emerging standards such as ebXML [1]. Software Agents retain the potential to structure, design and build e-commerce systems which require complex interactions between autonomous distributed components [13]. In particular, Agent-mediated e-commerce is concerned with providing agent-based solutions which support different stages of the trading processes in e-commerce such as needs identification, product brokering, merchant brokering, contract negotiation and agreement, payment and delivery, and service and evaluation [9]. Moreover, the distinctive capability of peculiar agents, called “mobile agents” [11], to move across a networked e-commerce environment can extend that support by enabling advanced e-commerce solutions such as location-aware shopping, mobile and networked comparison shopping, mobile auction bidding, and mobile contract negotiation. In addition, with respect to the traditional paradigms (C/S, REV, COD), the exploitation of mobile agents allows for conservation of bandwidth, reduction of latency, protocol encapsulation, asynchronous and autonomous distributed execution, dynamic adaptation, seamless integration of heterogeneous system, robustness and fault tolerance [11]. Although none of these strengths are unique to mobile agents, no competing technique shares all of them.

To date, a multitude of agent- and mobile agent-based e-commerce applications and systems have been developed [9], which basically allow for the creation of even complex e-Marketplaces on the Internet. However, to more effectively evaluate such solutions and, more generally, the benefits of using Agents to develop e-Marketplaces proper methodologies and tools are required. Such methodologies and tools should allow to validate, evaluate and compare the effectiveness and the efficiency of agent-mediated e-Marketplace models, mechanisms, policies and protocols before their actual implementation and deployment, saving development efforts and identifying better solutions.

In [7], an agent-based framework for e-commerce simulation games has been developed by using Zeus, a Java-based multi-agent system developed at the British Telecom Lab. Its goal is to evaluate through multi-player shopping games, in which agents represent sellers, buyers, brokers and services of various kinds, the potential consequences of novel combinations of market models, business strategies and new e-services. In [14], an infrastructure for Internet e-Marketplaces, based on Aglets mobile agents which supports real commercial activities carried out by consumers, agents and merchants, has been proposed. Its goals is not only to provide an advanced e-commerce service, but also to evaluate several dispatching models for mobile agents.

Although useful insights into new models and strategies can be gained by playing properly constructed games or by evaluating real applications, discrete event simulators are highly required to evaluate how these systems work on scales much larger than the scales achievable in games or in real applications where humans are involved. Nevertheless, few research efforts have been devoted to analysing agent-mediated e-Marketplaces by means of discrete event simulation.

This paper proposes an approach to the modelling and analysis of agent-based e-Marketplaces which centres on a Statecharts-based methodology for the simulation of mobile agent-based applications and systems [3]. The approach is exemplified by defining and simulating a consumer-driven e-marketplace model, inspired by the system presented in [14], which offers mobile agent-based services for searching and buying goods.

The remainder of the paper is organized as follows. In section 2, a Statecharts-based approach for modelling and simulating agent-based systems is presented. Section 3 shows the application of the approach for modelling a consumer-driven, mobile agent-based e-marketplace. In section 4 a simulation scenario of the defined e-Marketplace model is described and evaluated. Finally, conclusions are drawn.

2. An Approach for Modelling and Simulating Agent-based Systems

The proposed approach considers as the starting point a high-level model of the agent-based system that was previously obtained using agent-oriented methodologies [5] covering the phases of requirements capture, analysis and high-level design (see Fig. 1). This model can be expressed by a set of Agent Types (AT) which embody activity and offer services, and by a set of Logical Communication Links (LCL) among agent types which embody interaction protocols.

The approach [3] consists of three phases: Detailed Design, Coding and Simulation.

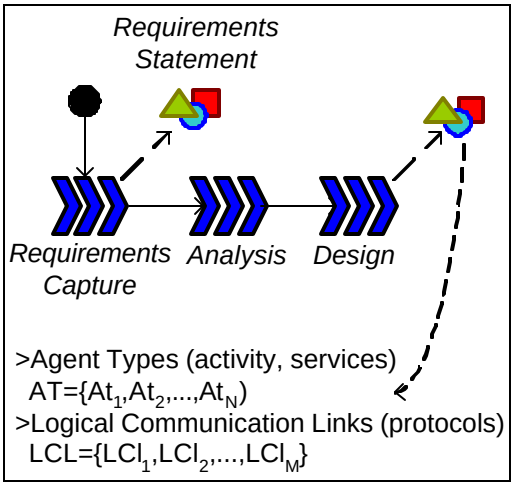


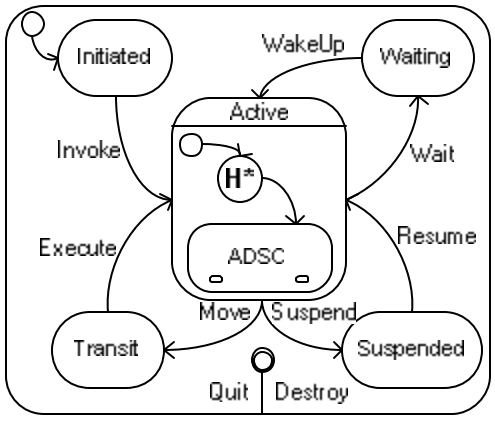
Figure 1. Requirements Capture, Analysis and High-Level Design.

A. Detailed Design

The d*etailed design* of the high-level model of the agent-based system is achieved through the visual specification of the behaviour of the agent types which embodies the definition of activity, services and interaction protocols. Visual specification is carried out using the Distilled StateCharts (DSC) formalism [4], derived from Statecharts [8], that allows for the modelling of the behaviour of lightweight agents, i.e. event-driven, single-threaded, capable of transparent migration, and executing chains of atomic actions.

The specification of the behaviour (*Abeh(At$_i$)*) of a lightweight agent type is carried out according to the FIPA-compliant agent behavioural template [6], reported in Fig. 2, which is a statechart consisting of a set of basic states (Initiated, Transit, Waiting, Suspended, and Active) and transitions labelled by events. In particular, an agent performs its computation and interaction activity in the ADSC (ACTIVE DSC) composite state, inside the Active state, which is to be refined by the agent designer. The presence of the deep history connector (H*) inside the Active state allows for the transparent migration of the lightweight agent as detailed in [4]. The *Abeh(At$_i$)* therefore consists of two parts: (i) a statechart (*Sbeh(At$_i$)*) which incorporates activity and interactions, and (ii) the related set of events (*Ebeh(At$_i$)*) to be handled which trigger state transitions in *Sbeh(At$_i$)*.

$$Abeh = \{Abeh(At_1), Abeh(At_2), \ldots, Abeh(At_N)\}$$
$$Abeh(At_i) = \langle Sbeh(At_i), Ebeh(At_i) \rangle$$



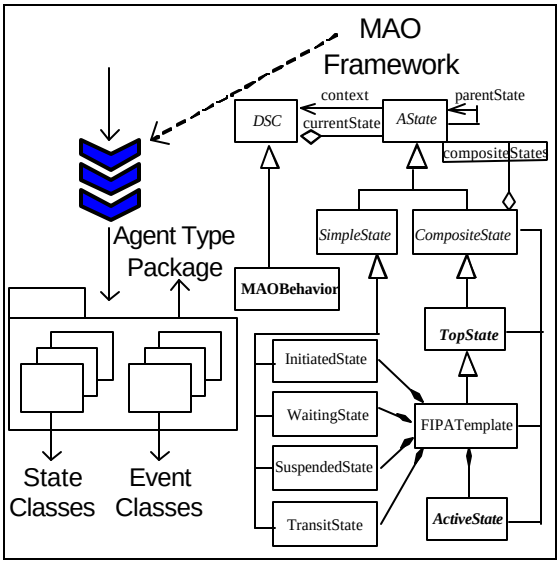


Figure 3. The Coding phase.

C. Simulation

The *simulation* of the agent-based system (see Fig. 4) is accomplished by means of a Java-based discrete event simulation framework for agent-based systems. Using this framework, an agent-based complex system can be easily validated and evaluated by defining a *simulator program* along with suitable test cases and performance measurements.

In particular, the *simulation engine* of the framework provides support for the: (i) execution of agents by interleaving their events processing, (ii) interchange of events among agents, (iii) migration of agents, and (iv) the clustering of agents into Agent Servers connected by a Logical Network.

The basic simulation entities offered by the framework are:

- the AgentServer, which is an object representing the agent server hosting mobile and stationary agents;
- the Agent, which is an object representing a stationary or a mobile agent (A_j) and including a pair of objects: $<Id_j, Abeh_j(At_i)>$, where Id_j is the agent identifier and $Abeh_j(At_i)$ is the *MAOBehavior object* related to the agent type At_i;
- the VirtualNetwork, which represents the logical network of hosts on which an AgentServer is mapped;
- the UserAgent, which is an object representing a user. A UserAgent, which is directly connected to an AgentServer, can create, launch and interact with Agents;
- the UserAgentGenerator, which is an object modelling the process of generation of a UserAgent.

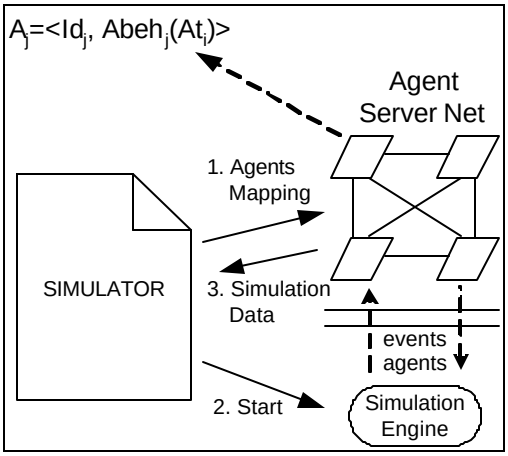


Figure 4. The Simulation phase.

In order to provide an high-level exemplification of how to construct a simulator without going into programming details, consider the following application scenario to analyze. A user is connected through a low-bandwidth link to a remote server which belongs to a fully-connected network of agent servers with higher-bandwidth links. The user sends a mobile agent to the connected remote server where the mobile agent is equipped with an itinerary (i.e. a set of agent server to be visited) provided by a stationary agent. The task of the mobile agent is to travel along the itinerary, perform some local computation and, finally, come back and report to its owner (i.e. the user created it). The purpose of the simulation is to analyze the average completion time of the mobile agent.

The simulator program can be constructed in the following steps:

(1) creation of the VirtualNetwork object which is constructed as a set of server nodes completely connected through high-bandwidth links;

(2) creation of the AgentServer objects and mapping of these objects onto distinct server nodes of the VirtualNetwork object;

(3) creation of the UserAgent object, which contains the code for the creation of the specific mobile agent (a purposely defined Agent object) and the code for the computation of the mobile agent completion time by marking the instants in time of the departure and arrival of the mobile agent, and binding of the UserAgent object to an AgentServer object through a low-bandwidth connection;

(4) creation of the stationary Agent object providing the itinerary and mapping of this object onto the AgentServer object to which the UserAgent object is bound;

(5) creation and insertion of the Invoke event directed to the UserAgent object into the event queue inside the simulation engine;

(6) initialization of the discrete-event clock and start of the simulation engine.

3. MODELLING AN AGENT-BASED ELECTRONIC MARKETPLACE

A consumer-driven e-Marketplace is an e-Marketplace in which the exchange of goods is driven by the consumers that wish to buy a product. The modelled agent-based e-Marketplace, inspired by the system presented in [14], consists of a set of both stationary and mobile agents which provides basic services for the buying and selling of goods.

Identification of the agent types along with their activity and of the logical communication links among the agent types along with their interaction patterns, was carried out by using Gaia [15].

Gaia is a methodology which has been

specifically tailored to the analysis and design of agent-based systems. It is founded on the view of a multi-agent system as a computational organisation composed of a number of autonomous interactive agents which play one or more specific roles. Gaia drives the designer of an agent-based application to obtain the aforementioned identification through the construction of the following set of models:

(i) the *Prototypical Roles Model*, the *Interactions Model* and the *Roles Model (analysis models), which* identify the roles occurring in the system and model interactions between the roles identified;

(ii) the *Agent Model*, the *Services Model* and the *Acquaintance Model (design models*), which, on the basis of the analysis models, define the types of agents in the system along with the services, the activities, and the logical communication paths of such agents.

Figure 5 reports the logical structure (or *acquaintance model*) of the agent-based e-Marketplace, highlighting the identified agent types and the logical communication links among them. In the following sections, the functionality of each agent type, the workflow of the system along with the interactions among the agent types, the different kinds of mobile consumer agents, and the DSC specification of a model of mobile consumer agent are illustrated.

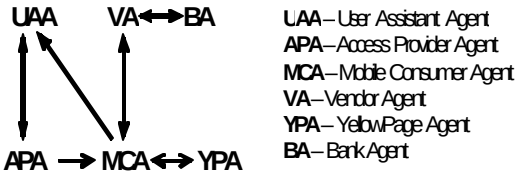


Figure 5. Logical structure of the agent-based e-Marketplace.

D. Types of Agents

User Assistant Agent (UAA). An UAA is associated with a user and assists her/him in: (i) looking for a specific product that meets her/his needs; (ii) buying the product according to a specific buying policy.

Access Provider Agent (APA). An APA represents the entry point of the e-Marketplace. It receives requests for buying a product from a registered UAA and fulfils them by generating a specific Mobile Consumer Agent (MCA).

Mobile Consumer Agent (MCA). A MCA is an autonomous mobile agent that deals with the searching, contracting, evaluation, and the payment of goods. In order to buy a product, a MCA is equipped with a wallet containing a limited amount of e-cash (or "bills").

Vendor Agent (VA). A VA represents the vendor of specific goods.

YellowPage Agent (YPA). A YPA represents an entry point of the federated yellow pages service (or "Yellow Pages") which provides the location of agents selling a given product. The following organizations of Yellow Pages were established:

- Centralized: each YPA stores a complete list of VA agents;
- One Neighbour Federated: each YPA stores a list of VA agents and keeps a reference to only one other YPA;
- M-Neighbours Federated: each YPA stores a list of VA agents and keeps a list of at most M YPA agents.

Bank Agent (BA). A Bank Agent represents a reference bank of MCA and VA agents. Bills owned by such agents are unique, cryptographically signed documents issued by one of the accredited banks. A bill can be represented by a few bytes of information containing: the name of the bank, the amount of the bill, the unique bill identifier, and the bank's signature needed to check the authenticity of the bill.

E. System Workflow

The system workflow is structured in the following phases:

1. *Request Input***.** When users wish to buy a product, they interact with their associated UAA which is delegated the buying task by specifying a set of buying parameters: product description, maximum price (P_{MAX}), Searching Policy (SP) and Buying Policy (BP). Users is notified by their UAA about the task results as soon as the task is completed. To perform its task an UAA contacts the APA with which it is

registered and submits a request containing the product parameters specified by the user. If the UAA is trustworthy (i.e. from a commercial and security viewpoint [14]), the APA accepts the request and creates a specific MCA by passing along the product parameters and the location of the initial YPA to be contacted.

2. *Searching*. The MCA obtains a list of locations of VA agents which sell the requested product by using the Yellow Pages. Searching can be carried out by adopting one of the following searching policies:
   - ALL: all YPA agents are contacted;
   - PARTIAL (PA): a subset of YPA agents are contacted;
   - ONE-SHOT (OS): only one YPA is contacted.
3. *Contracting & Evaluation.* The MCA interacts with the VA agents in the obtained list to request an offer for the desired product (P_{offer}), evaluates the received offers, and selects an offer, if any, for which the price is acceptable (i.e., $P_{offer}=P_{MAX}$) according to the following buying policies:
   - Minimum Price (MP): the MCA first interacts with all the VA agents to look for the lowest price of the product; then, it buys the product from the VA which offers it at the lowest acceptable price;
   - First Shot (FS): the MCA interacts with the VA agents until it obtains an offer for the product at an acceptable price; then, it buys the product;
   - Fixed Trials (FT): the MCA interacts with a given number of VA agents and buys the product from the VA which offers it at the lowest acceptable price;
   - Random Trials (RT): the MCA interacts with a random number of VA agents and buys the product from the VA which offers it at the lowest acceptable price.
4. *Payment.* The MCA moves to the location of the selected VA and pays the desired product using a given amount of bills. The following basic protocol is used to execute the money transaction between the MCA and the VA: (i) the MCA gives the bills to the VA; (ii) the VA sends the bills to its BA; (iii) the BA validates the authenticity of the bills, exempted them from re-use, and, finally, issues an amount of bills equal to that previously received to the VA; (iv) the VA notifies the MCA.
5. *Reporting.* The MCA reports the buying result to the UAA. On the basis of an unsuccessful buying result (*vendor not found*, *offers not acceptable*) the user can submit a new request either raising P_{MAX} or using a different combination of searching and buying policies.

F. Models of Mobile Consumer Agents

A behaviour model for the MCA can be defined on the basis of a tuple: <SP, BP, TM>, where SP is a searching policy in {ALL, PA, OS}, BP is a buying policy in {MP, FS, FT, RT}, and TM is a task execution model. Two different task execution models were defined:

- *Itinerary*: the Searching and Contracting & Evaluation phases are performed by a single MCA which fulfils its task by sequentially moving from one location to another within the e-Marketplace;
- *Parallel*: the Searching and Contracting & Evaluation phases are performed by a set of auto-coordinating mobile agents in a parallel way. The MCA is able to generate a set of children (generically called workers) and to dispatch them to different locations; the workers can, in turn, spawn other workers.

An MCA task execution model is chosen by the APA when it accepts a user input request; the choice can depend on the pair <SP, BP> selected by the user and on the e-Marketplace characteristics. If the chosen task execution model is of the *Parallel* type then the MCA is named PCA (*Parallel Consumer Agent*) otherwise if the chosen task execution model is of the *Itinerary* type then the MCA is named ICA (*Itinerary Consumer Agent*). Therefore, a *PCA model* is defined by a tuple <SP, BP, *parallel*> whereas an *ICA model* is defined by a tuple <SP, BP, *itinerary*>.

G. Programming the Mobile Consumer Agents

The DSC specification of the defined PCA model (or simply PCA) is reported in Fig. 6. The defined ICA model can be seen as a particular case of the PCA. With reference to Fig. 6, it is worth pointing out that:

- events are asynchronously received and processed according to a run-to-completion semantics (i.e. an event can be processed only if the processing of the previous event is fully completed);
- the received events can be asynchronously generated by the agent itself (internal events) or by other agents (external events) through the primitive
  generate(<mevent>(<param>)), where mevent is an event instance and param is the list of formal parameters of mevent including the identifiers of the event sender and of the event target, and (possibly) a list of event parameters.

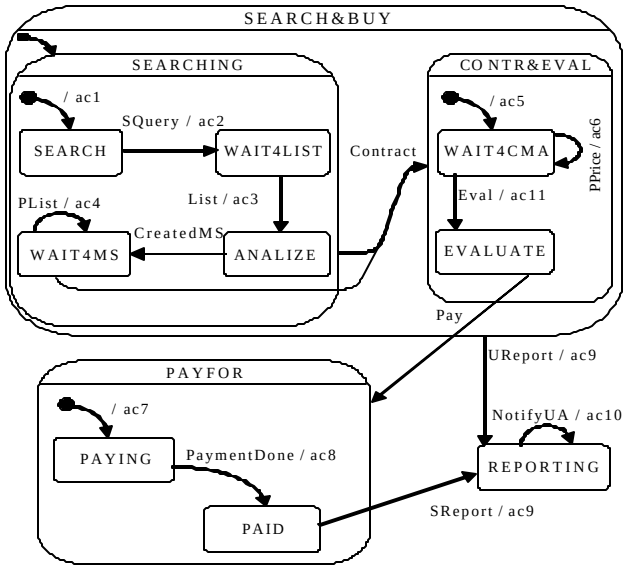
The PCA accomplishes the searching phase in the SEARCHING state.

In particular, as soon as the PCA is created, it moves (ac1) to the first YPA location and locally interacts (ac2) with the YPATarget by sending it the *VAListQuery* event.

The YPATarget replies to the PCA with the *List* event which can contain a list of VA agents with linked YPA agents.

After processing the reply (ac3), the PCA can do one of the following:

- create an Itinerary Searcher Mobile Agent (ISMA), which sequentially moves from one YPA location to another, if the Yellow Pages are of the One-Neighbour Federated type, and passes (ac4) into the contracting phase as soon as a *PList* event sent by the ISMA is received. The *PList* event contains the partial list of vendors collected by the spawned agent. It is worth noting that the adoption of the ISMA to carry out the searching task can improve efficiency since an ISMA is *lighter* than a PCA;



ac1 : generate(new Move(self(), YPATarget.getCurrLocation()));
generate(new SQuery(self()));
ac2 : generate(new VAListQuery(self(), YPATarget, vaListQuery));
ac3 : List reply = (List)mevent; proc = *processInitialYPAReply*(reply)
if (proc.createISMA()) sa1();
else if (proc.createSSMA()) sa2();
else if (proc.noVendors()) sa3();
else sa4();
sa1 : generate(new Create(self(), "ISMA", nextYPATarget));
generate(new CreatedMS(self()));
sa2 : for (int i=0; i<ypaList.size(), i++)
generate(new Create(self(), "SSMA", ypaList.elementAt(i)));
generate(new CreatedMS(self()));
sa3 : generate(new UReport(self()));
sa4 : generate(new Contract(self()));
ac4 : PList reply = (PList)mevent; res = *processMSReply*(reply);
if (res.contract()) sa4()
else if (res.noVendors()) sa3();
ac5: for (int i=0; i<vaList.size(), i++)
generate(new Create(self(), "CMA", vaList.elementAt(i)));
ac6 : PPrice offer = (PPrice)mevent; eval = *evaluateOffer*(offer);
if (eval.buy()) generate(new Eval(self()));
else if (eval.noBuy()) sa3();
ac7 : bills = *prepareBills*(price);
generate(new PayFor(self(), VATarget, bills));
ac8 : nbills = nbills - eval.price();
generate(new SReport(self()));
ac9 : generate(new Move(self(), self().getHomeLocation()));
generate(new NotifyUA(self()));
ac10: *reportTR*();
ac11: generate(new Move(self(), VATarget.getCurrLocation()));
generate(new Pay(self()));

ac = action chain
sa = sub-action

Figure 6. ADSC of the behaviour of the PCA.

- create M Spawning Searcher Mobile Agents (SSMAs), if the Yellow Pages are organised according to the M-Neighbours Federated type, and pass (ac4) into the contracting phase when all the *PList* events sent by the directly created SSMA agents are processed. In particular, an SSMA moves to the assigned YPA and, in turn, creates a child SSMA for each reachable YPA. This parallel searching technique generates a spawning tree with SSMA agents as nodes and rooted at the PCA. If an SSMA interacts with a YPA which has already been visited by an SSMA belonging to the same spawning tree, the YPA notifies the SSMA which comes back to its parent;
- directly pass into the contracting phase if the organization of the Yellow Pages is Centralized;
- report an unsuccessful search to the UAA.

The contracting phase accomplished in the CONTRANDEVAL state involves the creation of a Contractor Mobile Agent (CMA) for each VA in the vaList. Each CMA moves to the assigned VA location, contracts with the VA, and finally returns to the PCA location to report. The evaluateOffer method, which embeds the buying policy, evaluates the VA offers (*PPrice* events) reported by the CMA agents and generates (ac6) a decision about when and from which VA to purchase. In the PAYFOR state the PCA pays (ac7) the VA using the *PayFor* event which contains the bills.

After receiving the *PaymentDone* event, the PCA passes (ac8) into the REPORTING state from where it moves back (ac9) to the original APA location and finally reports (ac10) to its UAA.

4. SIMULATION OF THE AGENT-BASED ELECTRONIC MARKETPLACE

The primary goal of the simulation phase which was performed was to validate the defined agent-based e-Marketplace model and particularly:

(i) the behaviour of each type of agent,

(ii) the different models of MCA agents in each type of the Yellow Pages organizations, and

(iii) the agent interactions over the logical communication links.

The second goal of the performed simulation phase was to better understand the effectiveness of the simulation for evaluating the performances of different agent-mediated e-Marketplaces solutions. To this purpose, the completion time of the buying task was individuated as the main performance index. In particular, with reference to the proposed model, the completion time of ICA and PCA was evaluated.

The simulation and analysis parameters are presented in Table 1.

Table 1.
Simulation and Analysis parameters

N_{VA}	Number of VA agents
N_{YPA}	Number of YPA agents
YPO	Yellow Pages Organization type: {Centralized, 1-Neighbour, 2-Neighbour}
δ_{MA}	Link delay between two adjacent nodes for transmitting an agent
δ_{MSG}	Link delay between two adjacent nodes for transmitting a message
$T_C=T_{REPORT}-T_{CREATION}$	Completion time of the MCA, where $T_{CREATION}$ is the time of the MCA creation and T_{REPORT} is the time of the MCA report

The simulated e-Marketplace was set up as follows:

- each stationary agent (UAA, APA, YPA, VA, BA) executes in a different agent server;
- the agent servers are mapped onto different network nodes which are completely connected through links which have the same characteristics. The communication delay (d) on a network link is modelled as a lognormally distributed random variable with a mean, μ, and, standard deviation, s [2];
- each UAA is connected to only one APA;
- the price of a product, which is uniformly distributed between a minimum (PP_{MIN}) and a maximum (PP_{MAX}) price, is set in each VA at initialization time and is never changed; thus the VA agents adopt a fixed-pricing policy to sell products;
- each YPA manages a list of locations of VA agents selling available products.
- a UAA searches for a desired product, which always exists in the e-Marketplace, and is willing to pay a price P_{MAX} for the desired product which can be any value uniformly distributed between PP_{MAX} and $(PP_{MAX}+PP_{MIN})/2$.

In order to analyze e-Marketplaces having different structures and dimensions, the simulations were run by varying (i) the organization of the Yellow Pages (Centralized, 1-Neighbour and 2-Neighbour organized as a binary tree), (ii) the number of YPA agents in the range [10..1000] and (iii) the number of VA agents in the range [10..10000]. These ranges were chosen for accommodating small as well as large e-Marketplaces. The durations of the performed simulations were specifically set to allow for the completion of the buying task carried out by the MCA.

The results obtained from the simulations allowed to: (a) evaluate which task execution model is more appropriate with respect to SP and BP policies and for the characteristics of the e-Marketplace, and (b) validate the analytical model proposed in [14] regarding the sequential and parallel dispatching of mobile agents.

Regarding to point (a), the ICA performs better than the PCA in the following cases:

- SP={ALL, PA, OS}, BP=FS, YPO={Centralized, 1-Neighbour};
- SP={PA, OS}, BP=FS, YPO=2-Neighbour.

Thus, the APA can choose the itinerary task execution model if such cases occur.

Regarding to point (b), the performance evaluation focused on the <ALL, MP, *> models (see section 3.B) since they are the only models of MCA which guarantee both a successful purchase and the best purchase since they are successful at identifying the VA selling the desired product at the minimum price.

The results obtained for the <ALL, MP, * > MCA models over an YPA organization of the binary tree 2-Neighbour type are reported in Fig.7. The results shown in Fig. 7 were obtained with N_{YPA}={10, 100} and by varying N_{VA}. In agreement with the analytical model reported in [14], the PCA, due to its parallel dispatching mechanism, outperforms the ICA when N_{VA} and N_{YPA} increase.

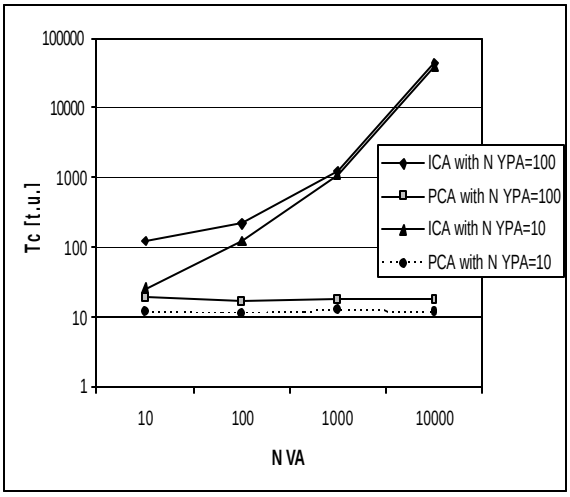


Fig. 7. Performance evaluation of the <ALL, MP, * > models for an e-Marketplace with YPO=2-Neighbour binary tree type, N_{YPA}={10, 100} and variable N_{VA}.

5. CONCLUSIONS

Flexible methodologies and tools for the modelling and simulation of agent-based systems are necessary to effectively support agent-oriented software development in complex application domains such as e-Commerce.

This paper has proposed a novel approach centred on Statecharts-based

tools to the modelling and analysis through simulation of agent-based systems. The approach has been exemplified by presenting a case study concerning with the modelling and the simulation of a consumer-driven e-Marketplace.

The approach provides the following valuable advantages:

(i) *Statecharts-based modelling language*. The use of a modelling language based on Statecharts, which are included in UML, reduces the learning curve for modelling due to the pervasive exploitation of UML in Industry and Academia;

(ii) *Validation through simulation*. The use of the simulation to validate agent-based systems before their actual deployment and execution, is strategic. In fact, the simulation, particularly if event-driven, is the only viable means to validate large-scale and complex systems.

In addition, the use of the MAO Framework allows for a seamless translation of the agent behaviour model into code, reducing the discontinuities between modelling and implementation phases.

Current research efforts are geared at modelling multiple integrated marketplaces and addressing through simulation security, efficiency and scalability issues of such large-scale multi agent systems.

REFERENCE

[1] ebXML, http://www.ebxml.org.

[2] Floyd, S., Paxson, V., "Difficulties in simulating the Internet," *IEEE/ACM Transactions on Networking 9(4)*, 2001, pp. 392-403.

[3] Fortino, G., Russo, W., "A Statecharts Based Methodology for the Simulation of Mobile Agents," *Proc. of the EUROSIS European Simulation and Modeling Conference (ESMc'03)*, Naples, 27-29, Oct., 2003, pp. 77-82.

[4] Fortino, G., Russo, W., Zimeo, E., "A Statecharts-based Software Development Process for Mobile Agents," *Information and Software Technology*, 46(13), pp. 907-921, 2004.

[5] Fortino, G., Garro, A., Russo, W., "From Modeling to Simulation of Multi Agent Systems: an Integrated Approach and a Case Study," Proc. of the 2nd Conference on Multi-Agent system TEchnologieS (MATES), Erfurt, Germany, Sept. 2004. to appear in Springer series of LNAI.

[6] Foundation of Intelligent and Physical Agents (FIPA), http://www.fipa.org.

[7] Griss, M., Letsinger, R., "Games at Work – Agent-Mediated E-Commerce Simulation," *Proc. of ACM Autonomous Agents*, Barcellona, Spain, Jun. 2000.

[8] Harel, D., Gery, E., "Executable Object Modelling with Statecharts," *IEEE Computer 30(7)*, 1997, pp. 31-42.

[9] Kowalczyk, R., Ulieru, M., Unland, R., "Integrating Mobile and Intelligent Agents in Advanced e-Commerce: A Survey," *Agent Technologies, Infrastructures, Tools, and Applications for E-Services*, NODe 2002 Agent-Related Workshops, Erfurt, Germany, October 7-10, 2002. Kowalczyk, R., Müller, J.P., Tianfield, H., Unland, R. (Eds.): LNCS 2592 Springer, 2003.

[10] Luck, M., McBurney, P., Preist, C., "Agent technology: enabling next generation computing: A roadmap for agent-based computing," *AgentLink report*, 2003. Available from www.agentlink.org/roadmap.

[11] Lange, D.B., Oshima, M., "Seven good reasons for Mobile Agents," *Communications of the ACM*, 42(3), 1999, pp. 88-89.

[12] Medjahed, B., Benatallah, B., Bouguettaya, A., Ngu, A.H.H., Elmagarmid, A.K., "Business-to-business interactions: issues and enabling technologies," *The VLDB Journal* , 12, 2003, pp. 59–85.

[13] Maes, P., Guttman, R.H., Moukas, A., "Agents that buy and sell: Transforming commerce as we know it," *Communications of the ACM*, 42(3), Mar. 1999, pp. 81-91.

[14] Wang, Y., Tan, K-L., Ren, J., "A Study of Building Internet Marketplaces on the Basis of Mobile Agents for Parallel Processing," *World Wide Web: Internet and Web Information Systems*, 5, 2002, pp. 41-66.

[15] Wooldridge, M., Jennings, N. R., Kinny, D., "The Gaia methodology for agent-oriented analysis and design," *Journal of Autonomous Agents and Multi-Agent Systems*, 3(3), 2000, pp. 285–312.