ELDAMeth: A Methodology For Simulation-based Prototyping of Distributed Agent Systems

Giancarlo Fortino and Wilma Russo
Department of Electronics, Informatics and Systems (DEIS)
University of Calabria
Via P. Bucci, cubo 41C, I-87036 Rende (CS), Italy
{g.fortino, w.russo}@unical.it

Abstract—**In application domains, such as distributed information retrieval, content management and distribution, e-Commerce, the agent-based computing paradigm has been demonstrated to be effective for the analysis, design and implementation of distributed software systems. In particular, several agent-oriented methodologies, incorporating suitable agent models, frameworks and tools, have been to date defined to support the development lifecycle of distributed agent systems (DAS). However, few of them provide effective methods for dynamic validation to analyze design objects at different degrees of refinement before their actual implementation and deployment. In this paper, ELDAMeth, a simulation-based methodology for DAS that enables rapid prototyping based on visual programming, automatic code generation and dynamic validation, is presented. ELDAMeth can be used both stand-alone for the modeling and evaluation of DAS and coupled with other agent-oriented methodologies for enhancing them with simulation-based validation. In particular, the proposed methodology is based on the ELDA (Event-driven Lightweight Distilled StateCharts-based Agents) agent model, and provides key programming abstractions (event-driven computation, multi-coordination, and coarse-grained strong mobility) very suitable for highly dynamic distributed computing and on a CASE tool-driven iterative process fully supporting the modeling, simulation, and implementation phases of DAS. A simple yet effective case study in the distributed information retrieval domain is used to illustrate the proposed methodology.**

Keywords – agent oriented software engineering; simulation; CASE tools; mobile agents; multi-coordination; statecharts

I. INTRODUCTION

The ubiquitous diffusion and usage of the Internet have promoted the development of new kinds of distributed applications characterized by a huge number of participants, high decentralization of software components and code mobility, which are typical of application domains such as distributed information retrieval, content management and distribution, and e-Commerce. In these application domains, the agent-based computing paradigm [19] has been demonstrated to be effective for the analysis, design and implementation of distributed software systems. In particular, in the context of the agent-oriented software engineering (AOSE), several agent-oriented methodologies based on suitable agent models, frameworks and tools, have been defined to support the development lifecycle of distributed agent systems (DAS). The key elements, identified through an in-depth analysis of such methodologies, for the provision of an effective development of distributed agent systems are the agent model, the development methodology and the supporting CASE tool.

The agent models aim at providing abstractions for the modelling of the agent behavior and interactions. Basically they can be classified in two large groups: (i) models based on intelligent agent architectures [19, 21] ranging from reactive agents (e.g. Brook's subsumption architecture) to deliberative agents (e.g. BDI agents); (ii) models based on the mobile active object concept encompassing mobile agent architectures [4]. Models of the first group are mainly oriented to problem-solving, planning and reasoning systems whereas models of the second group are more oriented to distributed computation in open and dynamic environments like the Internet. In the context of Internet computing, agent models and related frameworks based on lightweight architectures, asynchronous messages/events and state-based programming such as JADE [2], Bond [3], and Actors [1], have demonstrated great effectiveness for modeling and programming agent-based distributed applications. In particular, such models define suitable abstractions for the modelling of reactiveness and proactiveness of agent behaviors and interactions. However, they mainly consider messages (and related message-based protocols and infrastructures) as a means of interaction among agents and mobility as an auxiliary feature of agents. The exploitation of coordination models and infrastructures based not only on messages but also on events, tuples, blackboards and other coordination abstractions [6] can provide more effectiveness in designing complex agent interactions and more efficiency in their actual implementation. Moreover, mobility, which can provide a powerful means for dynamic organization of distributed components modeled as mobile agents, also enables and demands for new non-message-based coordination models.

The agent-oriented development methodologies aim at supporting the development lifecycle of agent-based systems from analysis to deployment and maintenance. They can be classified into general-purpose and domain-specific methodologies. The general-purpose methodologies such as Gaia [27], PASSI [7], Tropos [5], Ingenias [23] are suitable for the development of multi-agent systems in different application domains whereas the domain-specific methodologies can be

more effectively exploited in a given, very specific application domain. Apart from their context of use, they are all based on a meta-model of multi-agent system, which loosely or tightly depends on a reference agent model, and on a phase-based iterative development process. Agent oriented methodologies for Internet-based distributed agent systems should incorporate not only a MAS meta-model and its related agent model suitable for distributed computation but also effective prototyping methods able to validate the design models before their implementation and deployment in a large-scale distributed testbed. In particular, dynamic validation based on simulation is emerging as a powerful means for functional and non functional validation of designed agent systems in a large-scale controlled environment. To date a few agent-oriented, simulation-based development methodologies have been proposed in the literature, such as Electronic Institutions [26], DynDEVS/James [17], CaseLP [20], GAIA/MASSIMO [12], PASSIM [8], TuCSon/pi [16], Joint Measure [25], Ingenias/Repast [24]. They incorporate simulation to support the design phase of the MAS development lifecycle with the main focus on the validation and performance evaluation of the designed MAS model. Moreover, the importance of two additional features of agent-oriented methodologies, high degree of integration with other methodologies and availability of a CASE tool supporting the process phases, has become relevant in the AOSE community. The former feature would allow for an easy integration with other methodologies for the purpose of enriching already existing methodologies or creating new and more effective ones. The latter would allow for automating the development process phases and their transitions so providing more robust development and rapid prototyping.

In this paper we propose a novel methodology, named ELDAMeth, which provides all the aforementioned important features for the development of DAS: effective agent model for distributed computing systems, simulation-based agent-oriented methodology, integration with other methodologies, and CASE tool support. In particular, ELDAMeth relies on the ELDA (Event-driven Lightweight Distilled Statecharts Agents) agent model and related frameworks and tools, and on an iterative development process seamlessly covering the modeling, simulation and implementation phases of DAS and supported by a visual CASE tool. ELDAMeth can be used both stand-alone and in conjunction/integration with other agent-oriented methodologies which provide support to the analysis, (high-level) design, and implementation phases. ELDAMeth is exemplified through a case study concerning distributed information retrieval based on mobile agents.

The rest of the paper is organized as follows. Section II presents ELDAMeth, providing an overview of the modeling abstractions and tools, whereas the simulation phase of ELDAMeth is detailed in section III. In section IV the case study is described from modeling to simulation. Finally conclusions are drawn and future work anticipated.

II. ELDAMeth: A SIMULATION-BASED PROTOTYPING METHODOLOGY FOR DAS

ELDAMeth is a methodology specifically designed for the simulation-based prototyping of DAS. It is based on the ELDA (Event-driven Lightweight Distilled StateCharts Agent) agent model and related frameworks and tools, and on an iterative development process covering modeling, simulation and implementation phases of DAS. ELDAMeth can be used both *stand-alone* and in conjunction/integration with other agent-oriented methodologies which support the analysis and (high-level) design phases. In Figure 1, the development process of ELDAMeth is represented which consists of the following three phases:

- The *Modeling* phase produces an ELDA-based MAS design object that is a specification of a MAS fully compliant with the ELDA MAS meta-model. The design object can be produced either by (i) the ELDA-based modeler which uses the ELDA MAS meta-model and the ELDATool [11, 9], a CASE tool supporting the development phases of ELDA-based MAS, or by (ii) translation and refinement of design objects produced by other agent-oriented methodologies such as PASSI [7, 8], GAIA [27, 12], MCP [14], and others [23, 5, 19]. In particular, while the translation process centers on (semi) automatic model transformations based on the MAS meta-model of the employed methodology and the ELDA MAS meta-model, the refinement process is usually carried out manually by the ELDA-based Modeler by using the ELDATool. The defined design objects can be automatically translated, through the ELDATool, into ELDA-based MAS code objects according to the ELDAFramework, which is a set of Java classes formalizing all the modeling abstractions of the ELDA MAS meta-model. The code objects are then used in the *Simulation* phase.
- The *Simulation* phase produces the Simulation Results in terms of MAS execution traces and performance indices that must be carefully evaluated with respect to the identified functional and non-functional requirements. Such evaluation can lead to a further iteration step which starts from a new (re)modeling activity. In particular, the Simulation Results come from the execution of the ELDA-based MAS simulation object carried out through ELDASim, a Java-based event-driven simulation framework for ELDA agents. The simulation object is obtained by synthesizing the ELDA-based MAS code object with the simulation parameters and performance indices, defined on the basis of the requirements, by means of ELDASim.
- The *Implementation & Deployment* phase produces code for the JADE framework which can be then deployed and executed on a distributed JADE platform. Starting from the ELDA-based MAS design object, the code production is supported by the JADE-based DistilledStateChartBehaviour framework [15], the JADE framework and the ELDATool. Of course, the execution results can be evaluated against the functional and non functional requirements and, possibly, trigger a new iteration.

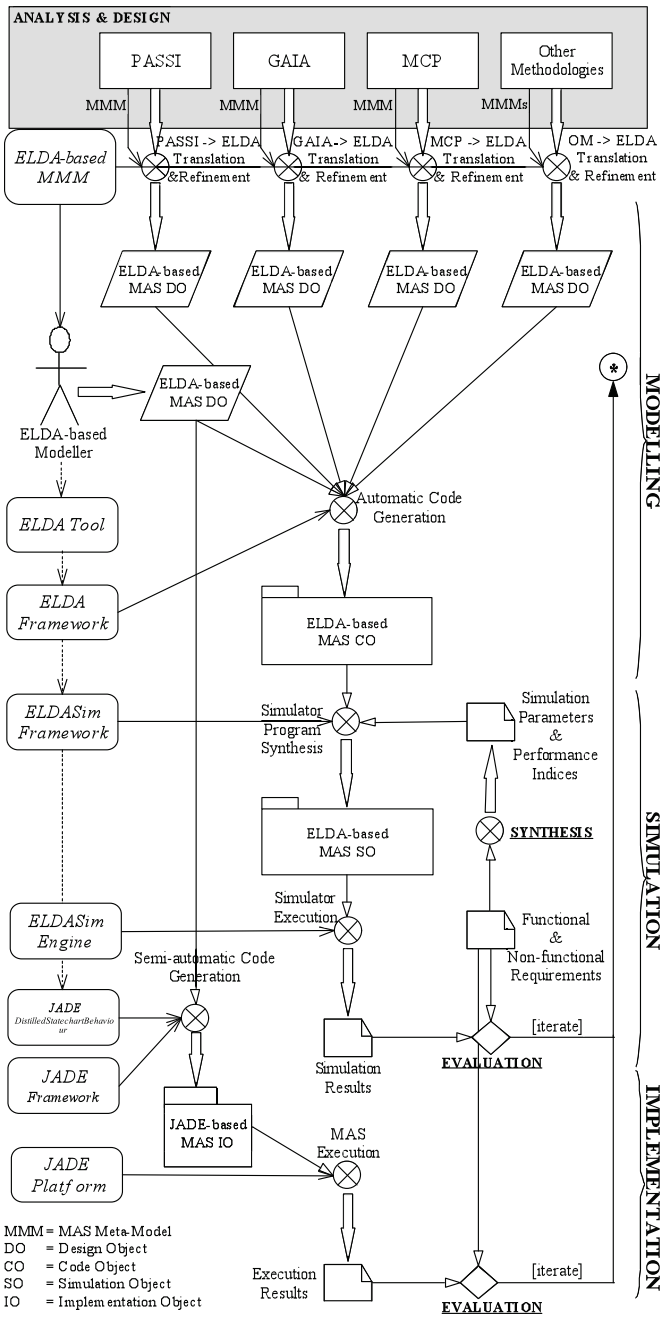

Figure 1. The ELDAMeth iterative development process.

In the following subsection a description of the ELDA-based modeling abstractions and tools is given (more details can be found in [13]) and the simulation phase is .

A. Modeling ELDA-based MAS

The modeling of agent-based systems based on the ELDA model is carried out through the ELDA MAS meta-model (ELDA MMM) which was specifically defined to provide design abstractions particularly suitable for DAS and specifically concerning agent lightweightness, multi-coordination and mobility. However, as agent-system domains could require specific design abstractions which haven't been originally included within the ELDA MMM, the structure of the ELDA MMM was designed to be extensible; in fact, ELDA MMM makes it possible to introduce new design abstractions (such as new services providers or new coordination spaces) which characterize a specific execution environment. In particular, the ELDA MMM is structured according to the view-based schema reported in Figure 2:

- *Agent View*, which represents the structure of an ELDA agent and its relationships with the coordination and system spaces. ELDA agents are event-driven lightweight agents that are a single-threaded autonomous entity interacting through asynchronous events, executing upon reaction, and capable of migration [13].
- *Event View*, which represents the structure of events. Events formalize both self-triggering events (Internal events) and requests to or notifications from the local agent server (Management, Coordination and Exception events). Events are further classified into OUT-events which are generated by the agent and always target the local agent server and IN-events which are generated by the local agent server and delivered to target agents.
- *SystemSpace View*, which represents the structure of the system space. The System Space provides system services for the management of agent lifecycles, timers and resources (e.g. consoles, databases, files, sensors). Agents interact with the System Space through Management events.
- *CoordinationSpace View*, which represents the hierarchy of the coordination spaces. The Coordination Space represents a local or global coordination structure based on a given coordination model through which agents interact. Several coordination spaces are currently defined such as message-based, local tuple space, publish/subscribe. Agents interact with the Coordination Space through Coordination events.
- *DSC View*, which represents the structure of a DSC, basically a hierarchical state machine with history pseudostates [13].
- *FIPATemplate View*, which represents the structure of the FIPA agent template [10] of the ELDA agent behavior.

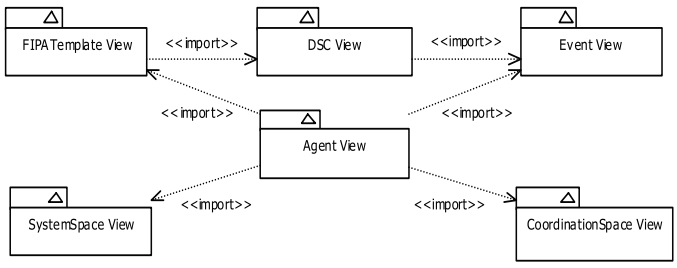

Figure 2. The ELDA MMM schema.

Models designed through the ELDA MMM can be coded through the ELDAFramework that is an object-oriented framework enabling developers to implement an ELDA-based application as it offers the implementation abstractions

representing the modeling concepts offered by the ELDA MMM.

To facilitate the use of ELDAMeth, an integrated development environment, named ELDATool [9, 11], is offered. It aims to support developers during the modelling, simulation, and implementation phases. In particular, ELDATool provides in an integrated fashion:

- a visual editor which allows to model behavior, interaction and mobility aspects of an agent-system according to the ELDA model;
- an automatic translator which implements the translation rules from ELDA meta-model to ELDAFramework;
- a visual editor to configure simulation parameters used to generate a simulation program based on ELDASim framework (see next sub section);
- an automatic translator which implements the translation rules from hybrid JADE and ELDA meta-models to the JADE DistilledStateChartsBehaviour.

The graphical design models are serialized into XML-like files. The tool also offers the functionality of automatic code generation by translating the XML-like files produced after the Modelling phase into Java code based on the ELDAFramework for simulation purposes or on the JADE framework for real execution.

Finally, to support the Simulation phase ELDATool offers a visual editor to configure simulation parameters which are used to generate the simulation program according to the ELDASim framework (see next subsection).

Currently, the ELDATool is implemented in Java as a collection of Eclipse plug-in to exploit several frameworks which fully support the development of visual editors; moreover, the high diffusion of Eclipse in the research community makes the tool immediately available to the Eclipse users and the learning process of the tool is therefore quicker.

B. Simulation of ELDA-based MAS

The development process of ELDAMeth includes a simulation phase (Figure 3) which consists of the following three activities:

1. *Performance Indices Definition*, which, on the basis of functional and non functional requirements, produces the definition of the performance indices which will be evaluated during the simulation;
2. *Simulation Implementation*, which aims at the realization of a simulation program which takes into account the previously identified indices, the definition of the controlled environment and the ELDAFramework-based DAS implementation. In particular, such program uses abstractions provided by the ELDASim (see below) to define:
   - the controlled execution environment (both features characterizing the computational nodes and the network) which mirrors the real execution environment;
   - the initial DAS configuration (agents and related locations);
3. *Simulation Execution*, which consists of the DAS execution within the controlled execution environment and of the collection of the defined performance indices which allow the analysis and the validation of the DAS under-development.

The simulation phase can be iteratively executed to modify, according to obtained simulation results, the modelling choices taken in former iterations. Simulation execution is supported by the ELDA simulation environment (ELDASim) which is a Java-based execution environment for ELDA agents that aims to validate and evaluate through simulation an ELDA model based solutions with respect to efficacy and efficiency aspects. To accomplish this, ELDASim is equipped with:

- The basics mechanisms of the distributed architectures supporting ELDA agents. In particular, agent servers, the network interconnecting agent servers, and several kinds of coordination infrastructures (asynchronous message-based, publish/subscribe, and tuple spaces) for fully supporting the distinctive multi-coordination feature of the ELDA model.
- The simulation of accomplishment time of time-consuming operations such as agent actions, agent management operations, coordination acts, and agent migrations.
- The capture of the traces of interactions (among agents and between agents and agent servers) in terms of exchanged events, filtered in an application-specific fashion.

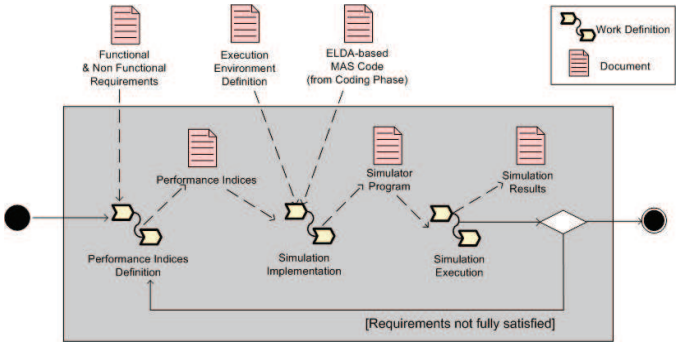

Figure 3. Schema of the Simulation phase.

III. A CASE STUDY: MOBILE AGENT-BASED DISTRIBUTED INFORMATION RETRIEVAL

In this section, a simple yet effective case study concerning with a distributed information retrieval task in a distributed computing system is proposed to exemplify ELDAMeth. In particular, the task consists in searching for specific information located exactly in one location within a network of federated information locations. The defined high-level solution is based on a coordinated set (or task force) of mobile agents which carry out the information searching task. A user (represented by an owner agent) starts searching by creating and launching a task force of mobile agents (called searcher agents) onto different random locations. As soon as the task force finds the desired information, the owner agent is notified

with the found information. A high-level design of such prototypical solution, which is to be properly translated and refined, is provided by the Multi-Coordination Process (MCP) [14]. In the following subsections the case study is described starting from the high-level modeling provided by MCP and then proceeding with the modeling, simulation and implementation phases.

1) MCP-based high-level modeling

The Multi-Coordination based Process (MCP) [14] is iterative and consists of the two phases (Modeling and Evaluation). The Modeling phase, on the basis of a coordination statement (CS) which derives from a preliminary analysis and includes a description of the agents along with their interactions (coordination requirements - CRs), and a set of coordination properties (CPs), provides alternative coordination solutions which fulfill the CS. In the Evaluation phase, a specific solution is chosen among such alternative coordination solutions which are evaluated through simulation and then compared on the basis of ad-hoc defined performance indices (e.g. time and resource consumption).

With reference to the case study, the proposed solutions for the coordination of the task force during its information retrieval task is based on the following CRs:

- CR_1: every time a searcher agent visits a location not yet searched by other agents of the same task force, it notifies the other agents that such location has already been searched so avoiding unnecessary and resource-consuming duplicate searches;
- CR_2: as soon as a searcher agent finds the desired information on a given location, it reports the found information to the owner agent;
- CR_3: when a searcher agent finds the desired information on a given location, it signals such event to all the other searcher agents to stop them;

and on the following CPs:

- CP_a: the task force is constituted by at least two searcher agents;
- CP_b: the agents of the task force may or may not know each other whereas they know the identity of the owner agent and vice-versa;
- CP_c: the interactions among all the agents (searcher and owner) are always asynchronous.
- CP_d: the interactions required by CR_1 may be local or remote, that required by CR_2 and CR_3 are remote.

The defined solutions are reported in Figure 4. For each solution, the three coordination requirements are addressed by suitable interaction patterns (IPs) and related coordination models (CMs).

CR	IP	CM	IMPLEMENTATION DESCRIPTION
CR_1	LBN [2..N, *known*, local, async]	QAMP	When a searcher agent searches in a location which has not been already searched by another agent of its task force, it creates a marker agent, stationing in this location, which can locally signal the other agents of its task force that a search has already carried out. As soon as an agent visits a location looks for a marker agent of its task force to avoid searching.
CR_2	R2O [2, *known*, remote, async]	QAMP	When a searcher agent finds the desired information, it sends a message containing the found information to its owner.
CR_3	GBN [2..N, *known*, remote, async]	QAMP	A searcher agent which has found the desired information sends a notification message to all the other searcher agents of the task force to stop them.

(A)

CR	IP	CM	IMPLEMENTATION DESCRIPTION
CR_1	GBN [2..N, *known*, remote, async]	QAMP	A searcher agent to notify that it has searched a given location sends a message containing the location identifier to all the other searcher agents of the task force.
CR_2	R2O [2, *known*, remote, async]	QAMP	When a searcher agent finds the desired information, it sends a message containing the found information to its owner.
CR_3	GBN [2..N, *known*, remote, async]	QAMP	A searcher agent which has found the desired information sends a notification message to all the other searcher agents of the task force to stop them.

(B)

CR	IP	CM	IMPLEMENTATION DESCRIPTION
CR_1	LBN [2..N, un*known*, local, async]	LTS	When a searcher agent searches in a location which has not been already searched by another agent of its task force, it inserts a signaling tuple into the LTS to signal that this location has been searched. As soon as an agent visits a location and reads the signaling tuple, it avoids searching.
CR_2	R2O [2, *known*, remote, async]	QAMP	When a searcher agent finds the desired information, it sends a message containing the found information to its owner.
CR_3	GBN [2..N, *known*, remote, async]	TPS	When a searcher agent finds the desired information, it publishes an event of a specific topic related to its task force which signals the stop of the retrieval task. All the other agents of the task force will be thus asynchronously notified since they subscribed to the specific topic at creation time.

(C)

CR	IP	CM	IMPLEMENTATION DESCRIPTION
CR_1	GBN [2..N, un*known*, remote, async]	TPS	A searcher agent to notify that it has searched a given location publishes an event of a specific topic related to its task force and containing the location identifier. All the other agents of the task force will be thus asynchronously notified since they subscribed to the specific topic at creation time.
CR_2	R2O [2, *known*, remote, async]	QAMP	When a searcher agent finds the desired information, it sends a message containing the found information to its owner.
CR_3	GBN [2..N, *known*, remote, async]	TPS	When a searcher agent finds the desired information, it publishes an event of a specific topic related to its task force which signals the stop of the retrieval task. All the other agents of the task force will be thus asynchronously notified since they subscribed to the specific topic at creation time.

(D)

Figure 4. The result of the high-level MCP-based modeling: solutions A, B, C, D.

The IPs are selected from a repository containing some of the most used agent-oriented interaction patterns and specifically characterized (through the tuple [number of participants, participant identity, locus, temporality]) according to the coordination requirements and properties. In particular, the characterized IPs are:

- Location-based notification (LBN), which involves agents passing through a given location to be notified about events occurring/occurred in such location.
- Report to owner (R2O), which involves a child agent reporting to its owner agent when its task is completed.
- Group-based notification (GBN), which involves an agent notifying all agents of its group when a given event occurs.

The CMs used to implement the IPs are:

- Local Linda-like tuple space (LTS), which supports a high number of participants, allows temporal decoupling but only offers local interaction [22].
- Topic-based publish/subscribe (TPS), which supports a high number of participants, allows for distributed interactions and does not require temporal coupling between participants [18].
- Queue–based unicast asynchronous message passing (QAMP), which supports a variable number of participants, allows for both local and remote interactions, does not require temporal coupling, but requires spatial coupling among participants [28].

2) ELDA-based Modeling

The four solutions have been modeled according to the ELDA MMM and integrated into an ELDASim-based simulator program. In Figure 5 the SearcherAgent behavior of the solution A (see Fig.4A) is shown; such solution is representative of the message-based solution, whereas the fully multi-coordinated solution can be found in [13].

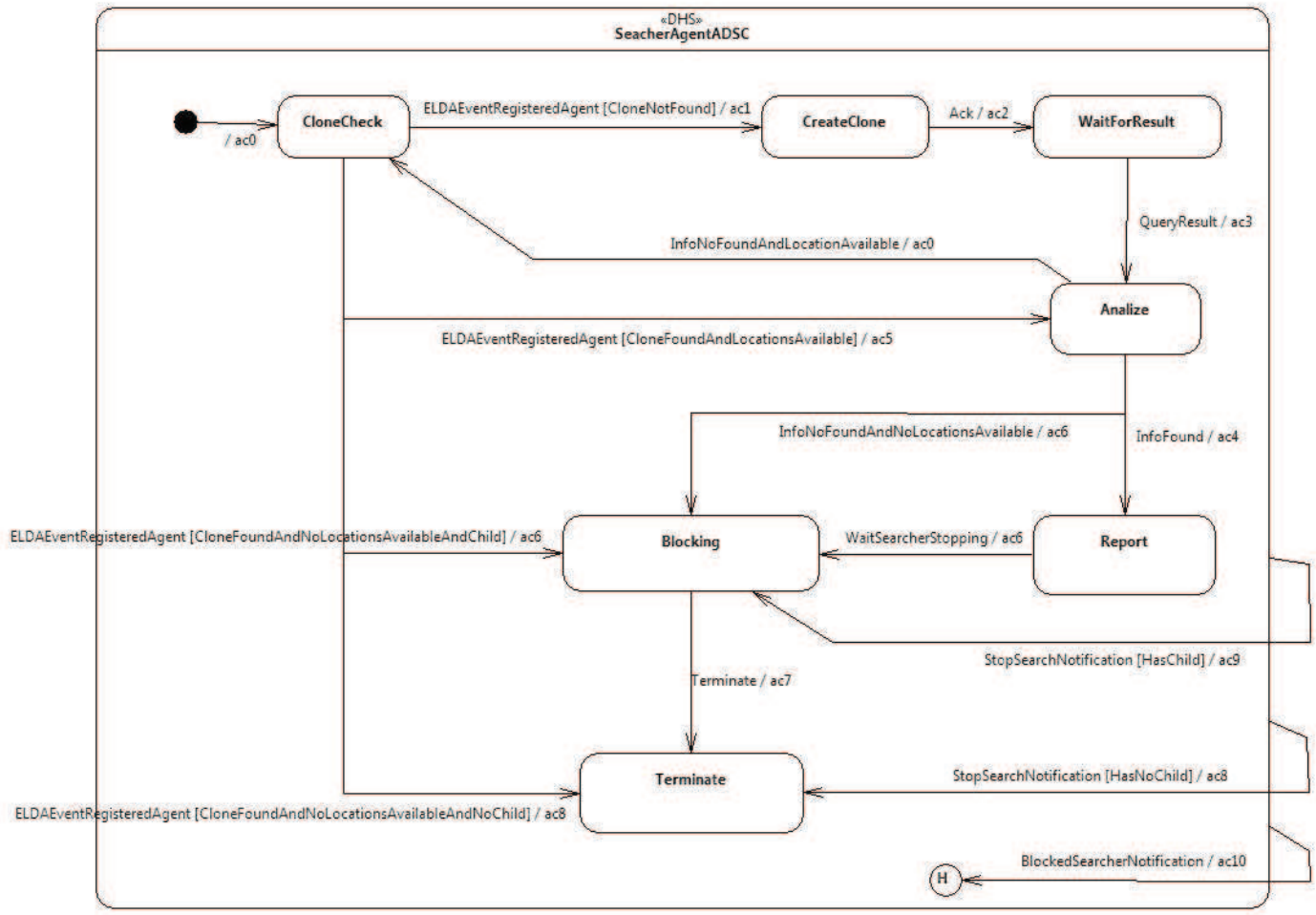

Figure 5. SearcherAgent behavior of solution A

Every time a Searcher Agent (SA) visits a new location, it checks for the presence of a marker agent by using the whitepage service made available by the agent server (action ac0) and behaves as follows:

1. If no marker agent is present, the SA creates the marker agent (action ac1), submits the query to the servant agent

(action ac2), and waits for the query result to analyze it (action ac3). If no info is found the SA moves to a new location (if available); otherwise the SA goes into a pseudo-termination state (BLOCKING). If the info is found, the SA notifies the user agent and the other members of the task force through asynchronous messages (action ac4); then, it goes into the BLOCKING state. The user agent will thus receive a Report event whereas the taskforce members the StopSearchNotification event. Upon reception of a StopSearchNotification event, a SA stops its activity and goes into the BLOCKING state if it has previously created marker agents; otherwise, it terminates. In both cases, such SA will notify the other taskforce members of its state change (actions ac8 and ac9). Going into the BLOCKING state, the SA notifies its state change to the other taskforce members (action 6) so that each member knows the active agents in the taskforce. The notification is based on the BlockedSearcherNotification event that, once received, allows updating the list of active agents (action ac10). Moving from BLOCKING to TERMINATED the SA requests to cease its activity and also sends to all the marker agents it has previously created a termination request (action ac7).

2. If the marker agent is present and other locations are available, the SA migrates to a new location (action ac5).
3. If the marker agent is present and no other locations are available but the SA has created at least one marker agent in previously visited locations, it goes into the BLOCKING state (action ac6) to enforce the termination of such marker agents (see point 1 for the management of marker agent termination).
4. If a marker agent is present, no other locations are available and no marker agents have been previously created by the SA, the SA terminates (action ac8).

3) Simulation and performance evaluation

To evaluate the four solutions shown in Figure 4, in the *Performance Indices Definition* activity, the performance indices reported in Table 1 have been defined. The *Simulation Execution* activity relies on two simulation parameters (the number of locations and the number of *searcher agent*s) and on the following settings of the network topology and information distribution:

- Locations are connected through a fully connected logical network composed of FIFO channels. In particular, channels are characterized by the same delay and bandwidth parameters modeled as uniform random variables.
- The information to be found is contained exactly at one location and the locations keep references (randomly generated) to other locations at information level to be all reachable.

Simulation runs are carried out with the number of locations equals to 100 and the number of searcher agents in the range [2..20]. Moreover, for each simulation run, all four solutions are executed on the same network topology and information distribution. In Figures 6-10 the simulation results are reported; the obtained values of the performance indices are averaged over 50 simulation runs.

TABLE I. PERFORMANCE INDICES

Task Completion Time (T_{TC})	The duration between the spawning of the first created *searcher agent* and the first report message received by the *owner agent.*
Number of Messages (N_M)	The number of coordination messages transmitted by the agents across the network.
Notification Time (T_N)	The duration between the information finding and the notification to the last *searcher agent.*
Number of Visits (N_V)	The total number of locations visited by the *searcher agent*s after the information finding.
Number of Searches (N_S)	The total number of the locations searched by the *searcher agent*s after the information finding.

The T_{TC} performance index, which measures the speed with which the information search task is carried out, decreases as the number of searcher agents increases (see Figure 6). In fact, the use of more searcher agents augments the degree of parallelism which, consequently, increases the probability to find the searched information with a smaller number of migrations which are time-consuming. The performances of all the solutions are almost the same.

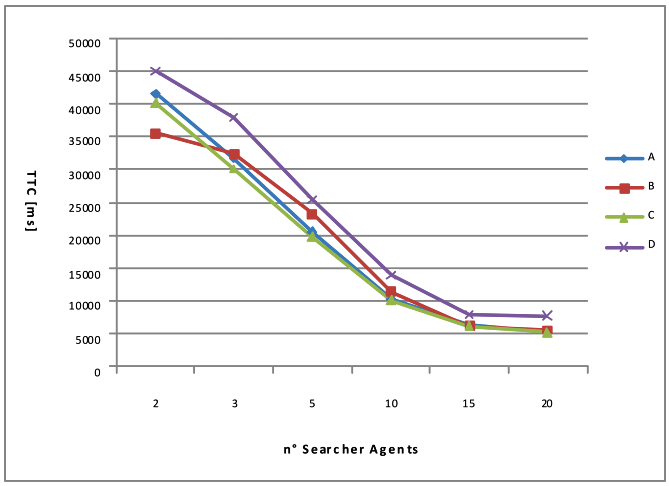

Figure 6: The Task Completion Time

The N_M parameter (see Figure 7), which measures the network load, is significantly better in the A and C solutions thus saving network resources with respect to the other solutions. In fact, as CR_1 is modeled according to the LBL interaction patterns in solutions A and C whereas the GBN interaction pattern is used in solutions B and D the number of coordination messages increases due to the GBN interaction pattern is adopted by B and D. The T_N performance index measures how fast all the searcher agents are notified after finding the information: the shorter T_N, the fewer are the resources consumed throughout the agent platform. The A and

C solutions outperform the other solutions (see Figure 8) as the network load is lighter than the ones of the B and D solutions. The N_V and N_S parameters are measures of the consumption of resources after the information is found. The values of such parameters should be kept as low as possible. As shown in Figures 9 and 10, the A and C solutions outperform the other solutions also for such indices. In particular, although the T_{TC} values of the A and C solutions are similar to the solutions B and D, the other performance indices values are significantly better. It is worth noting that solution A not only is less straightforward than solution D but also requires the cloning of a marker agent for each visited location but such operation may not be allowed according to security policies of the locations..

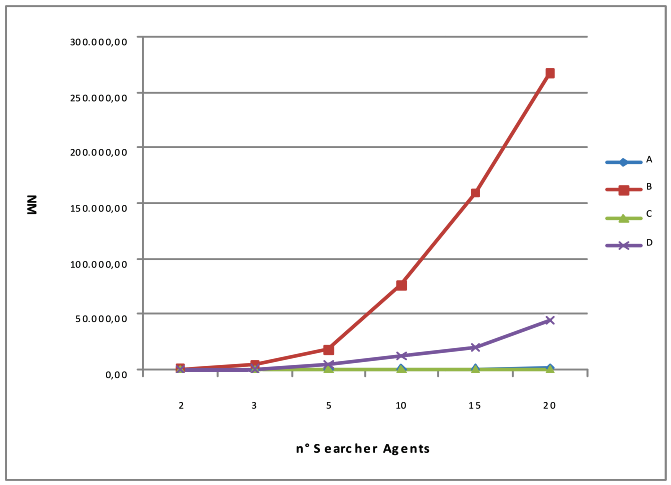

Figure 7: The Number of coordination messages

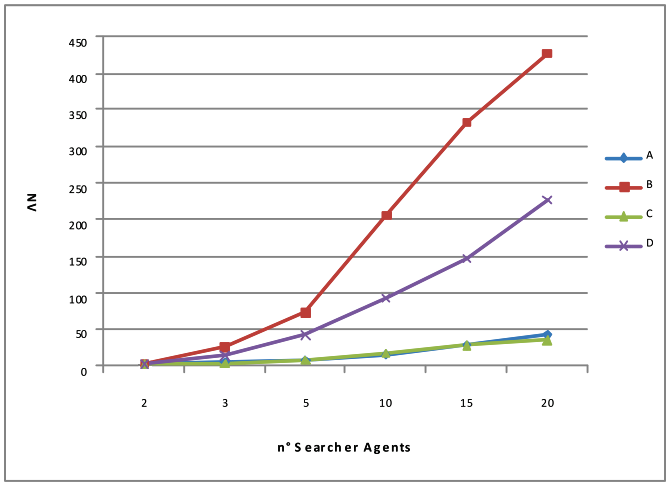

Figure 9: The Number of visits after finding information

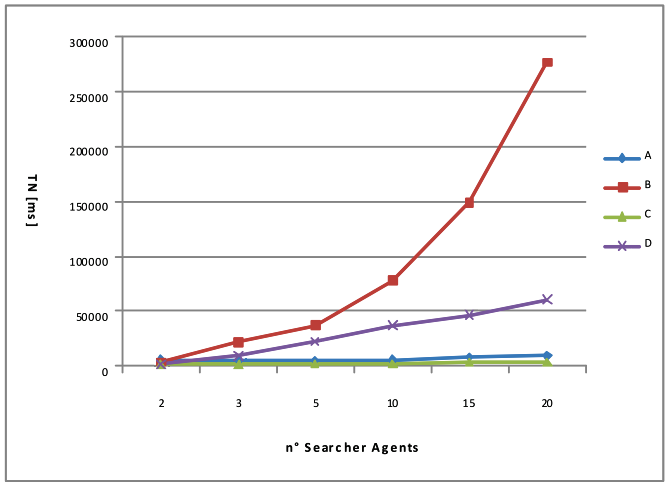

Figure 8: The Notification Time

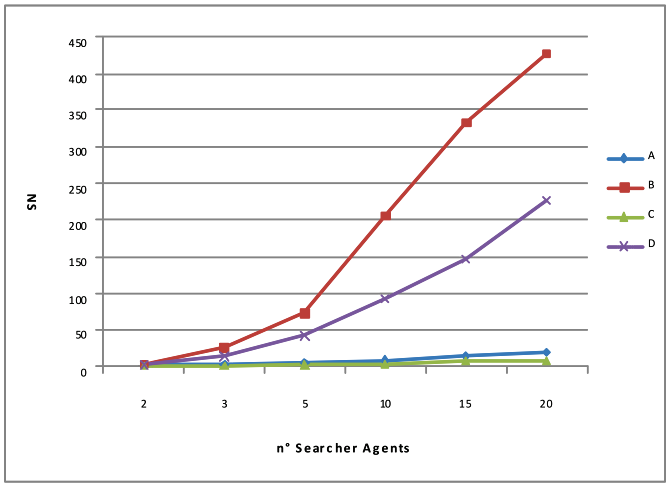

Figure 10: The Number of searches after finding information

IV. CONCLUSION

This paper has proposed ELDAMeth, a novel agent-oriented methodology supported by a CASE tool for the simulation-based prototyping of Internet-based distributed agents systems (DAS). In particular, the distinctive characteristics of ELDAMeth are: effective agent model for distributed computing systems, simulation-based agent-oriented methodology for design validation before implementation and deployment, integration with other methodologies to exploit their well-defined method fragments, and CASE tool support for supporting all development phases from modeling to simulation and implementation. Such distinctive characteristics make ELDAMeth very effective for prototyping Internet-oriented DAS. ELDAMeth has been applied to prototype several kinds of DAS such as mobile e-Marketplaces, content delivery infrastructures, and information retrieval systems. In this paper we have shown a case study in the information retrieval domain which has demonstrated the suitability and great effectiveness of ELDAMeth for the rapid prototyping of Internet-based DAS. As future work we aim at providing full support to multi-coordination in the implementation phase: this would allow to translate multi-coordinated ELDA specifications into a real target platform represented by JADE and coordination infrastructures such as TucSon for tuple spaces and Elvin for publish/subscribe systems.

ACKNOWLEDGMENT

Authors wish to thank A. Garro, S. Mascillaro, G. Mazzitelli, and F. Rango for useful ideas, discussions and implementation efforts supporting the ELDAMeth project.

REFERENCES

[1] Astley, M., and Agha, G. A., Customization and Composition of Distributed Objects: Middleware Abstractions for Policy Management, ACM SIGSOFT 6th International Symposium on Foundations of Software Engineering (FSE), 1998.

[2] Bellifemine, F., Poggi, and A., Rimassa, G. 2001. Developing multi agent systems with a FIPA-compliant agent framework. Software Practice And Experience 31, 103-128.

[3] Boloni, L., and Marinescu, D. C., A multi-plane state machine agent model, Fourth International Conference on Autonomous Agents, Barcelona, Spain, pp. 80-81, ACM Press, 2000.

[4] Braun, P. and Rossak, W., Mobile Agents: basic concepts, mobility models, & the tracy toolkit, Heildelberg, Germany, Morgan Kaufmann Publisher, 2005.

[5] P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos, and A. Perini. Tropos: An Agent- Oriented Software Development Methodology. Journal of Autonomous Agents and Multi- Agent Systems. Kluwer Academic Publishers Volume 8, Issue 3, Pages 203 - 236, May 2004.

[6] Cabri, G., Leonardi, L.., and Zambonelli, F., Mobile-agent coordination models for internet applications, IEEE Computer, 33, 2, pp 82-89, 2000.

[7] Cossentino, M., From Requirements to Code with the PASSI Methodology, Agent-Oriented Methodologies, B. Henderson-Sellers and P. Giorgini (eds). Idea Group Inc., Hershey, PA, USA, 2005.

[8] Cossentino, M., Fortino, G., Garro, A., Mascillaro, S. and Russo, W. 2008. PASSIM: a simulation-based process for the development of multi-agent systems. Int. J. Agent-Oriented Software Engineering 2(2), 132-170.

[9] ELDATool documentation and software, http://lisdip.deis.unical.it/software/eldatool.

[10] FIPA Agent Management Specification, Management for agents on FIPA agent platforms, http://www.fipa.org/specs/fipa00023/SC00023K.html.

[11] Fortino, G., Garro A., Mascillaro S., and Russo W. 2007. ELDATool: A Statecharts-based Tool for Prototyping Multi-Agent Systems. In Proceedings of Workshop on Objects and Agents (WOA'07, Genova, IT, Sept. 24-25, 2007), pp. 14-19.

[12] Fortino, G., Garro, A., and Russo, W. 2005. An Integrated Approach for the Development and Validation of Multi Agent Systems. Computer Systems Science & Engineering 20, 4, 94-107.

[13] G. Fortino, A. Garro, S. Mascillaro, W. Russo, "Using Event-driven Lightweight DSC-based Agents for MAS Modeling," in the special issue "Best of From Agent Theory to Agent Implementation 6 (AT2AI-6)", International Journal on Agent Oriented Software Engineering (Inderscience publisher), 4(2), 2010.

[14] G. Fortino, A. Garro, S. Mascillaro, W. Russo, "A Multi-Coordination based Process for the Design of Mobile Agent Interactions," In Proceedings of IEEE Symposium on Intelligent Agents (IEEE Symposium Series on Computational Intelligence), Nashville (TN), USA, March 30-April 2, 2009.

[15] G. Fortino, F. Rango, W. Russo, "Statecharts-based JADE agents and tools for engineering Multi-Agent Systems", in Proc of 14th International Conference on Knowledge-Based and Intelligent Information and Engineering Systems (KES2010), Cardiff, 2010.

[16] L. Gardelli, M. Viroli, M. Casadei, A. Omicini, "Designing Self-Organising Environments with Agents and Artifacts: A Simulation-Driven Approach", International Journal of Agent-Oriented Software Engineering (IJAOSE). Volume 2(2). Page 171--195. 2008.

[17] Himmelspach J, Röhl M & Uhrmacher AM (2008): Component based models and simulation experiments for multi-agent systems in James II. In the Proc. of the 6th Int'l Workshop "From Agent Theory to Agent Implementation" (AT2AI) jointly held with AAMAS", Estoril, Portugal, 13 May, 2008.

[18] Loke, S. W., Padovitz, A., Zaslavsky, A., and Tosic, M., Agent Communication Using Publish-Subscribe Genre: Architecture, Mobility, Scalability and Applications, Annals of Mathematics, Computing & Teleinformatics, 1, 2, pp 35-50, 2004.

[19] Luck, M., McBurney, P., and Preist, C. 2004. A manifesto for agent technology: towards next generation computing. Autonomous Agents and Multi-Agent Systems 9, 3, 2004, 203-252.

[20] Martelli M., Mascardi, V. and Zini, F., Specification and Simulation of Multi-Agent Systems in CaseLP, Appia-Gulp-Prode Joint Conf. on Declarative Programming, L'Aquila, Italy. pp. 13-28, 1999.

[21] Nwana, H.S., Software Agents: an overview, Knowledge Engineering Review, 11, 3, pp 205–244, 1996.

[22] A. Omicini, F. Zambonelli, "Tuple Centres for the Coordination of Internet Agents," ACM Symposium on Applied Computing (SAC'99), 28 February - 2 March 1999.

[23] Pavón, J., Gómez-Sanz, J.J. and Fuentes, R. (2005) 'The INGENIAS methodology and tools', in B. Henderson-Sellers and P. Giorgini (Eds.) Agent-Oriented Methodologies, Idea Group Publishing, pp. 236-276.

[24] Pavon, J., Sansores, C., and Gomez-Sanz, J. J. 2008. Modelling and simulation of social systems with INGENIAS. *Int. J. Agent-Oriented Softw. Eng.* 2, 2 (Feb. 2008), 196-221.

[25] Sarjoughian, H.S., Zeigler, B.P. and Hall. S.B., A Layered Modeling and Simulation Architecture for Agent-based System Development, IEEE, 89, 2, pp. 201-213, 2001.

[26] Sierra, C., Rodríguez-Aguilar, J. A., Noriega, P., Esteva, M. and Arcos, J.L., Engineering Multi-agent Systems as Electronic Institutions, Novática, 170, 2004.

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

[28] Zhou, X.Y., Arnason, N., and Ehikioya, S.A., A proxy-based communication protocol for mobile agents: protocols and performance, IEEE Conference on Cybernetics and Intelligent Systems, vol. 1, pp 53-58, 1-3, Dec. 2004.