A Survey on Parallel and Distributed Multi-Agent Systems

Alban Rousset, Bénédicte Herrmann, Christophe Lang, and Laurent Philippe

Femto-ST Institute
University of Franche-Comté
16 Route de Gray 25030 Besançon cedex - France
{alban.rousset,bherrman,clang,lphilipp}@femto-st.fr

Abstract. Simulation has become an indispensable tool for researchers to explore systems without having recourse to real experiments. Depending on the characteristics of the modeled system, methods used to represent the system may vary. Multi-agent systems are, thus, often used to model and simulate complex systems. Whatever modeling type used, increasing the size and the precision of the model increases the amount of computation, requiring the use of parallel systems when it becomes too large. In this paper, we focus on parallel platforms that support multi-agent simulations. Our contribution is a survey on existing platforms and their evaluation in the context of high performance computing. We present a qualitative analysis, mainly based on platform properties, then a performance comparison using the same agent model implemented on each platform.

Keywords: multi-agent simulation, parallelism, MAS.

1 Introduction

In the field of simulation, we often seek to exceed limits, that is to say analyse larger and more precise models to be closer to the reality of a problem. Increasing the size of a model has however a direct impact on the amount of needed computing resources and centralised systems are often no longer sufficient to run these simulations. The use of parallel resources allows us to overcome the resource limits of centralised systems and also to increase the size of the simulated models.

There are several ways to model a system. For example, the time behavior of a large number of physical systems is based on differential equations. In this case the discretization of a model allows its representation as a linear system. It is then possible to use existing parallel libraries to take advantage of many computing nodes and run large simulations. On the other hand it is not always possible to model any time dependent system with differential equations. This is for instance the case of complex systems. A complex system is defined in [25] as "*A system that can be analyzed into many components having relatively many relations among them, so that the behavior of each component depends on*

L. Lopes et al. (Eds.): Euro-Par 2014 Workshops, Part I, LNCS 8805, pp. 371–382, 2014.

the behavior of others". Thus the complexity of the dependencies between the phenomena that drive the entities behavior makes it difficult to define a global law that models the entire system. For this reason multi-agent systems are often used to model complex systems because they rely on an algorithmic description of agents that interact and simulate the expected behavior. From the viewpoint of increasing the size of simulations, multi-agent systems are constrained to the same rules as other modelling techniques but there exists less support for parallel execution of the models.

In this article, we focus on multi-agent platforms that provide parallel distributed programming environments for multi-agent systems. Recently, the interest for parallel multi-agent platforms has increased. This is because parallel platforms offer more resources to run larger agent simulations and thus allows to obtain results or behavior that was not possible to obtain with smaller number of agents (eg. simulation of individual motions in a city/urban mobility).

The contribution of this article is a survey on parallel distributed multi-agent platforms. This survey is based on an extensive bibliographical work done to identify the existing platforms, a qualitative analysis of these platforms in terms of ease of development, distribution management or proposed agent model, and a performance evaluation based on a representative model run on a HPC cluster.

The article is organised as follows. First, we give the context of multi-agent system (MAS) in general and parallel distributed multi-agent systems (PDMAS) in particular. We then introduce the different multi-agent platforms found in our bibliographical research. In the third section, we describe the method used to classify platforms and we describe the model implemented in each platform to evaluate its performance. In the fourth section, we present the qualitative comparison of the different PDMAS followed by the benchmark based on the implemented model. We finish the paper with conclusion and future work.

2 Related Works

The concept of agent has been studied extensively for several years and in different domains. It is not only used in robotics and other fields of artificial intelligence, but also in fields such as psychology [6] or biology [23]. One of the first definitions of the agent concept is due to Ferber [13] :

> "An agent is a real or virtual autonomous entity, operating in a environment, able to perceive and act on it, which can communicate with other agents, which exhibits an independent behavior, which can be seen as the consequence of his knowledge, its interactions with other agents and goals it need to achieved".

A multi-agent system, or MAS, is a platform that provides the mandatory support to run simulations based on several autonomous agents. These platforms implement functions that provide services such as agent life cycle management, communication between agents, agent perception or environment management. Among well known platforms we can cite Repast Simphony [21], Mason [19],

NetLogo [28] and Gama [1]. These platforms however do not natively implement a support to run models in parallel and it is necessary to develop a wrapper from scratch, in order to distribute or parallelize a simulation. There exists several papers that propose survey on these multi-agent platforms [29,5,4,16].

Some platforms like RepastHPC [10], D-Mason [12], Pandora [2], Flame [8] or JADE [3] provide a native support for parallel execution of models. This support usually includes the collaboration between executions on several physical nodes, the distribution of agents between nodes and so on. During our analysis of the literature, we did not find any survey about parallel multi-agent platforms except the paper written by Coakley and al. [8]. This comparison is based on qualitative criteria such as the implementation language but the paper does not provide any performance comparison of the studied platforms.

After an extensive bibliographical work, we identified 10 implementations or projects of parallel multi-agent platforms. For each platform we tried to download the source or executable code and we tried to compile it and test it with the provided examples and templates. Some of the platforms cannot be included in our study because there is no available source code or downloadable executable (MACE3J [15], JAMES[17], SWAGES [24]), or because only a demonstration version is available (PDES-MAS [22,27]), or because there is a real lack of documentation (Ecolab [26]). It was thus not possible to build a new model in these platforms and thus to assess their parallel characteristics and performance. These platforms have subjected to a qualitative analysis which is not included in this paper.

For the 5 remaining platforms, on which we were able to implement our model, we can consider that they truly offer a functioning parallel multi-agent support. We succinctly present each of these platforms in the following.

D-Mason (Distributed Mason) [12] is developed by the University of Salerno. D-Mason is the distributed version of the Mason multi-agent platform. The authors choose to develop a distributed version of Mason to provide a solution that does not require users to rewrite their already developed simulations and also to overcome the limitations on maximum number of agents. D-Mason uses ActiveMQ JMS as a base to implement communications. D-Mason uses the Java language to implement the agent model.

Flame [8] is developed by the University of Sheffield. Flame was designed to allow a wide range of agent models. Flame provides specifications in the form of a formal framework that can be used by developers to create models and tools. Flame allows parallelization using MPI. Implementing a Flame simulation is based on the definition of X-Machines [9] which are defined as finite state automata with memory. In addition, agents can send and receive messages at the input and the output of each state.

Jade [3] is developed by the Telecom laboratory of Italia. The aims of Jade are to simplify the implementation of distributed multi-agent models across a FIPA compliant [3] middleware and to provide a set of tools that support the debugging and the deployment phases. The platform can be distributed across multiple computers and its configuration can be controlled from a remote GUI.

Agents are implemented in Java while the communications relay on the RMI library.

Pandora [2] is developed by the Supercomputing center of Barcelona. It is explicitly programmed to allow the execution of scalable multi-agent simulations. According to the literature, Pandora is able to treat thousands of agents with complex actions. Pandora also provides a support for a geographic information system (GIS) in order to run simulations where spatial coordinates are used. Pandora uses the $C++$ language to define and to implement the agent models. For the communications, Pandora automatically generates MPI code from the Pandora library.

RepastHPC [10] is developed by the Argone institute of USA. It is a part of a series of multi-agent simulation platforms: RepastJ and Repast Simphony. RepastHPC is specially designed for high performance environments. RepastHPC use the same concepts as the core of RepastSimphony, that is to say it uses also the concept of projections (grid, network) but this concept is adapted to parallel environments. The $C++$ language is used to implement an agent simulation but the ReLogo language, a derivative of the NetLogo language, can also be used. For the communications, the RepastHPC platform relays on MPI using the Boost library [11].

From these descriptions we can note that some platforms have already been designed to target high performance computing systems such as clusters whereas others are more focused on distribution on less coupled nodes such as a network of workstations.

3 Survey Methodology

In this section we explain the methodology used to make this survey. As already stated we started by a bibliographical search (using keywords on search engines and following links cited in the studied articles). This bibliographical search allowed us to establish a first list of existing platforms. By testing the available platforms we established a second list of functioning platforms. To our knowledge this list is complete and their is no other available and functional platform that provide a support for parallel distributed MAS. Note we only concentrate on distributed platforms and that the list excludes shared memory parallel platforms and many-cores (as GPU or Intel Xeon Phi) platforms. After we defined different criteria to compare and analyse each platform. We finished by implementing a reference model on each platform and executing it in order to compare the platform performance. These evaluation steps are detailed in the following.

This survey mainly focuses on the implementation, more precisely the development, of models and their execution efficiency. To classify the platforms we defined two sets of criteria: first, implementation and execution based criteria and, second, criteria about classical properties of parallel systems. We briefly explain in which correspond each criteria.

For the implementation and execution criteria, all platforms have their own constraints that impact on the ease of the model implementation. The chosen criteria are:

1. Programming language,
2. Agent representation
3. Simulation type, time-driven or event-driven
4. Reproductibility, do several executions of a simulation give the same results?

For the classical properties of parallel systems, we focus on:

1. Scalability of platform, in terms of agents and nodes,
2. Load balancing, agent distribution,
3. MultiThread execution, to take benefit of multicore processors,
4. Communication library.

To further compare the platforms, we have defined a reference agent model that we implemented on each platform. The reference model is based on three important behaviors for each agent: the agent perception, the communications between agents and/or with the environment and agent mobility. The reference model simulates each of these behaviors.

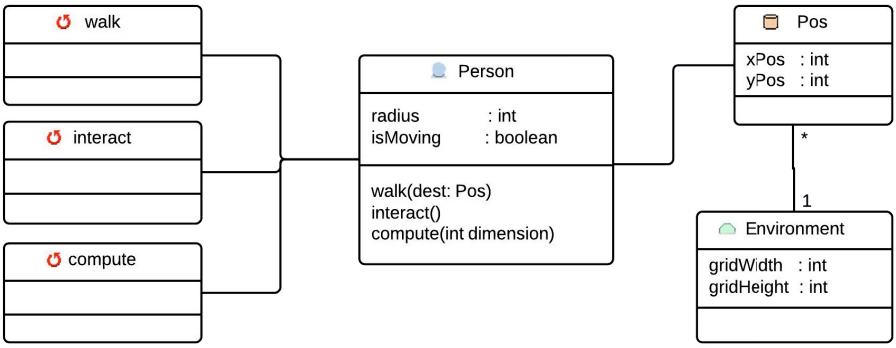

Fig. 1. AML representation of the reference agent model

Figure 1 gives an AML [7] (Agent Modeling Language) representation of our reference model. The *Environment* is represented by a square grid. *Agents* are mobile and move randomly on the grid. A *vision* characterised by the "radius" property is also associated with each agent. It represents the limited perception of the agent on the environment.

Each agent is composed of 3 sub-behaviors :

1. The walk behavior allows agents to move in a random direction on the environment. This behavior is used to test the mobility and the perception of the agents. As the agents walk through their environment to discover other agents and other parts of the environment, interactions and communications with the environment are also tested with this behavior.

2. The interact behavior allows agents to interact and send messages to other agents in their perception fields. This behavior intends to simulate communications between agents and to evaluate the communication support of the platforms.
3. The compute behavior allows agents to compute a "Fast Fourier Transform (FFT)" [14] in order to represent a workload. This behavior intends to simulate the load generated by the execution of the agent inner algorithms.

The global agent behavior consists in performing each of this three behaviors at each time step. The reference model has several parameters that determine the agent behavior and also the global model properties. For instance, the model allows to vary the workload using different sizes of input for the FFT calculus. It is also possible to generate more or less communications between agents by setting the number of contacted agents in the interact behavior or to assess the agent mobility by setting the agent speed in the walk behavior.

4 Qualitative Analysis

In this section we expose two levels of comparisons between the studied platforms: first a qualitative comparison using the previously presented criteria and second a performance comparison using the reference model.

Table 1 gives a synthetic representation of the comparison for the implementation and execution criteria. Most platforms use classical languages such as C-C++ or Java to define agents, except the Flame platform which uses the XMML language. The XMML language is an extension of the XML language designed to define X-Machines. Note that the RepastHPC platform implements, in addition to the C++ programming language, the widespread Logo agent language. The Repast-Logo or R-Logo is the Repast implementation of Logo for C++. It allows to simplify the simulation implementation at the price of a lower power of expression compared to C++.

Table 1. Comparison of implementation and execution properties

	RepastHPC	D-Mason	Flame	Pandora	Jade
Prog. lang.	C++/R-Logo	Java	XMML/C	C/C++	Java
Agent represent.	Object	Object	X-Machine	Object	Object
Simu. type	event-driven	time-driven	time-driven	time-driven	time-driven
Reproductibility	Yes	Yes	No	Yes	No

Agents are usually defined as objects with methods representing behaviors. An agent container gathers all the agents. This container is cut and distributed in the case of parallel execution. The agent implementation is different for the Flame platform that does not use the object concept to define a agent but rather uses automatas called X-Machines. In a X-Machine, a behavior is represented by a state in the automata and the order of execution between behaviors are

represented by transitions. This difference changes the programming logic of a model but induces no limitation compared with other platforms because agents are in fact encoded in C language.

For the simulation type, event or time driven, all platforms use the time-driven approach except RepastHPC which is based on the event-driven approach. RepastHPC however allows to fix a periodicity to each scheduled event, so that we can reproduce the behavior of time-driven simulations.

Finally all platforms allow agents to communicate. This communication can be performed either *internally* with agents that are on the same node, or *externally*, with agents that are on different nodes. The D-Mason and Pandora platforms propose remote method invocations to communicate with other agents while the other platforms use messages to communicate between agents.

Table 2 summarises the criteria of the platforms about classical properties of parallel systems. Globally we can note that all studied platforms meet the demands for the development of parallel simulations. Note that we did not find any information on the scalability property of the Pandora and Jade platforms, so they are marked as Not Available (NA) for this property. To efficiency exploit the power of several nodes the computing load must be balanced among them. There is different ways to balance the computing load . The load can be balanced at the beginning of the simulation (Static) or adapted during the execution (Dynamic). A dynamic load balancing is usually better as it provides a better adaptation in case of load variation during the model execution, but it can also be subject to instability. Most platforms use dynamic load balancing except the Jade and Flame platforms. In [20] the authors propose a way to use dynamic load balancing with the Flame platform.

Table 2. Comparison classical properties of parallel systems

	RepastHPC	D-Mason	Flame	Pandora	Jade
Scalability	1028 proc. [18]	36 nodes [8]	432 proc. [8]	NA	NA
Load Balancing	Dynamic	Dynamic	Static [8]	Dynamic	Static [3]
Multithread exec	Yes [8]	Yes [12,8]	No [8]	Yes	Yes
Com. library	MPI [11,10]	JMS [12]	MPI [18]	MPI [2]	RMI

Note that only Flame does not support multi-threaded executions. The platform however relays on the MPI messaging library. As most MPI libraries provide optimised implementations of message passing functions when the communicating processes are on the same node, using processes located on the same node instead of threads does not lead to large overhead. In the implementation of a multi-agent system this probably leads to equivalent performance as the simplification of synchronisation issues may compensate the cost of using communication functions.

Last, the communication support for most platforms is MPI. This is not surprising for platforms targeting HPC systems as this library is mainly used on these computers. Note that the D-Mason platform relays on the JMS communication service despite it is not the most scalable solution for distributed

environments. An MPI version of D-MASON is in development. Finally, the Jade platform is based on the java Remote Method Invocation (RMI) library which is not very adapted to parallel applications as it is based on synchronous calls. During the model implementation we also noted that the Jade platform seems to be more oriented for equipment monitoring and cannot be run on HPC computers due to internal limitations. Jade is thus not included in the rest of the comparisons.

5 Performance Evaluation

For the performance evaluation we have implemented the reference model defined in section 3 on the four functional platforms: RepastHPC, D-MASON, Flame, Pandora. During this model implementation, we did not encounter noticeable difficulties expect with the RepastHPC platform for which we have not been able to implement external communications, communications between agents running on different nodes. RepastHPC does not have the native mechanisms to make it whereas it is possible to implement it on the other platforms. RepastHPC actually offers the possibility to interact with an agent on an other node but not to report the modifications.

Although we have been able to run the four platforms, D-Mason, Flame, Pandora, RepastHPC, on a standard workstation, only two of them (RpastHPC, Flame) have successfully run on our HPC system. The D-Mason platform uses a graphical interface that cannot be disconnected. We are thus not able to run D-MASON on our cluster, only accessible through its batch manager. The Pandora simulations have deadlock problems even if we use examples provided with the platform. For these reasons the presented results only consider the Flame and RepastHPC platforms.

We have realised several executions in order to exhibit the platform behaviors concerning scalability (Figures 2 and 3) and workload (Figure 4). To assess scalability we vary the number of nodes used to execute the simulations while we fix the number of agents. We then compute the obtained speedup. For workload we fix the number of nodes to 8 and we vary the number of agents in the simulation. Each execution is realised several times to assess the standard variation and the presented results are the mean of the different execution durations. Due to a low variation in the simulation runtime, the number of executions for a result is set to 10.

About the HPC experimental settings, we have run the reference model on a 764 cores cluster using the SGE batch system. Each node of the cluster is a bi-processors, with Xeon E5 (8*2 cores) processors running at 2.6 Ghz frequency and with 32 Go of memory. The nodes are connected through a non blocking DDR infinyBand network organised in a fat tree. The system is shared with other users but the batch system garanties that the processes are run without sharing their cores.

Execution results for scalability for a model with 10 000 agents are given on Figure 2 and 3, with the ideal speedup reference. Note that the reference time

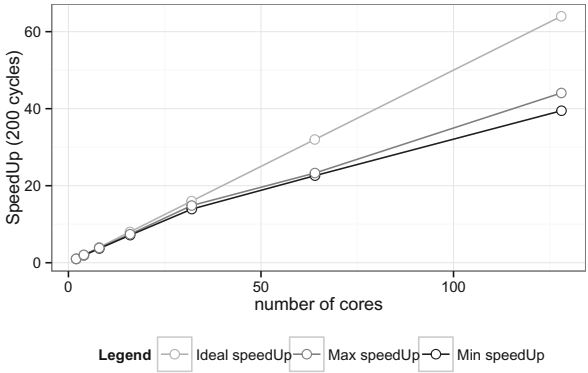

Fig. 2. Scalability of FLAME simulations using 10 000 agents, FFT 100 and 200 cycles

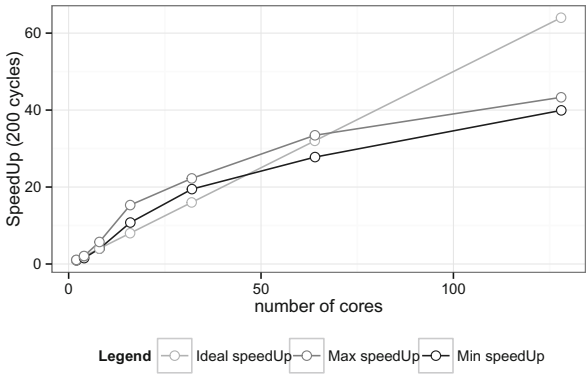

Fig. 3. Scalability of RepastHPC simulations using 10 000 agents, FFT 100 and 200 cycles

used to compute the speedup is based on a two core run of the simulations. This is due to RepastHPC which cannot run on just one core so that its reference time must be based on two core runs. The speedup is therefore limited to half the number of nodes. We can note that both platforms scale well up to 32 cores but the performance does not progress so well after, becoming 2/3 of the theoretical speedup for 128 cores. In addition on Figure 3 we can see that RepastHPC results are above the theorical speedup for simulations with less than 50 cores. As we suspected that these better results come from cache optimizations in the system, we did more tests to confirm this hypothesis. The realized tests increase the number of agents and the load on each agent to saturate the cache and force memory accesses. As the results for these new tests are under the theorical speedup the hypothesis is validated.

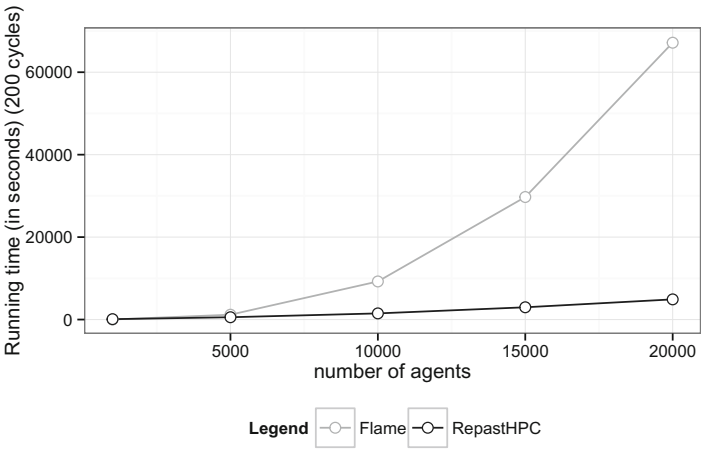

Fig. 4. Workload behavior for simulation using 8 cores

Figures 4 represents the workload behavior of the two platforms. The inner load of agents (FFT) is here set to 100. The figure shows that RepastHPC really better reacts to load increasing than Flame. The same behavior has also been noted for a load of 10 (for 20 000 agents the ratio is 0.92). On the opposite for a load of 1000 the difference is less noticeable (for 20 000 agents the ratio is 0.81). Obviously the used model does not use all the power of Flame as it is limited in term of inter-agent communications. The question to answer is: is it due to the use of the concept of X-Machines or synchronisation mechanisms in the underlying parallelism? Another possible reason that could justify this difference is the cost of the synchronisations provided by Flame when using remote agents and that is not managed in RepastHPC.

6 Conclusion

In this article we have presented a comparison of different parallel multi-agent platforms. This comparison is performed at two levels, first at a qualitative level using criteria on the provided support, and second at a quantitative level, using a reference agent model implementation. The qualitative comparison shows the properties of all the studied platforms. The quantitative part shows an equivalent scalability for both platforms but better performance results for the RepasHPC platform.

When implementing our reference model we have noticed that the synchronisation support of the platforms does not provide the same level of service: the RepastHPC platform does not provide communication support for remote agents while Flame do it. This support seems to be a key point in the platform performance.

For this reason, in our future work, we intend to better examine the efficiency of synchronisation mechanisms in parallel platforms. For example how are the

synchronizations made during an execution and is there a way to improve synchronization mechanisms in parallel multi-agent systems?

Acknowledgement. Computations have been performed on the supercomputer facilities of the Mésocentre de calcul de Franche-Comté.

References

1. Amouroux, E., Chu, T.-Q., Boucher, A., Drogoul, A.: GAMA: An environment for implementing and running spatially explicit multi-agent simulations. In: Ghose, A., Governatori, G., Sadananda, R. (eds.) PRIMA 2007. LNCS, vol. 5044, pp. 359–371. Springer, Heidelberg (2009)
2. Angelotti, E.S., Scalabrin, E.E., Ávila, B.C.: Pandora: a multi-agent system using paraconsistent logic. In: Computational Intelligence and Multimedia Applications, ICCIMA 2001, pp. 352–356. IEEE (2001)
3. Bellifemine, F., Poggi, A., Rimassa, G.: Jade–a fipa-compliant agent framework. In: Proceedings of PAAM, London, vol. 99, p. 33 (1999)
4. Berryman, M.: Review of software platforms for agent based models. Technical report, DTIC Document (2008)
5. Bordini, R.H., Braubach, L., Dastani, M., El Fallah-Seghrouchni, A., Gomez-Sanz, J.J., Leite, J., O'Hare, G.M., Pokahr, A., Ricci, A.: A survey of programming languages and platforms for multi-agent systems. Informatica (Slovenia) 30(1), 33–44 (2006)
6. Carslaw, G.: Agent based modelling in social psychology. PhD thesis, University of Birmingham (2013)
7. Červenka, R., Trenčanský, I., Calisti, M., Greenwood, D.P.A.: AML: Agent modeling language toward industry-grade agent-based modeling. In: Odell, J.J., Giorgini, P., Müller, J.P. (eds.) AOSE 2004. LNCS, vol. 3382, pp. 31–46. Springer, Heidelberg (2005)
8. Coakley, S., Gheorghe, M., Holcombe, M., Chin, S., Worth, D., Greenough, C.: Exploitation of hpc in the flame agent-based simulation framework. In: Proceedings of the 2012 IEEE 14th Int. Conf. on HPC and Communication & 2012 IEEE 9th Int. Conf. on Embedded Software and Systemsm, HPCC 2012, pp. 538–545. IEEE Computer Society, Washington, DC (2012)
9. Coakley, S., Smallwood, R., Holcombe, M.: Simon Coakley, Rod Smallwood, and Mike Holcombe. Using x-machines as a formal basis for describing agents in agent-based modelling. Simulation Series 38(2), 33 (2006)
10. Collier, N., North, M.: Repast HPC: A platform for large-scale agentbased modeling. Wiley (2011)
11. Collier, N.: Repast hpc manual (2010)
12. Cordasco, G., De Chiara, R., Mancuso, A., Mazzeo, D., Scarano, V., Spagnuolo, C.: A Framework for Distributing Agent-Based Simulations. In: Alexander, M., et al. (eds.) Euro-Par 2011, Part I. LNCS, vol. 7155, pp. 460–470. Springer, Heidelberg (2012)
13. Ferber, J., Perrot, J.-F.: Les systèmes multi-agents: vers une intelligence collective, InterEditions Paris (1995)
14. Frigo, M., Johnson, S.G.: The design and implementation of fftw3. Proceedings of the IEEE 93(2), 216–231 (2005)

15. Gasser, L., Kakugawa, K.: Mace3j: fast flexible distributed simulation of large, large-grain multi-agent systems. In: Proceedings of the First Inter. Joint Conf. on Autonomous Agents and Multiagent Systems: part 2, pp. 745–752. ACM (2002)
16. Heath, B., Hill, R., Ciarallo, F.: A survey of agent-based modeling practices (january 1998 to july 2008). JASSS 12(4), 9 (2009)
17. Himmelspach, J., Uhrmacher, A.M.: Plug'n simulate. In: Proceedings of the 40th Annual Simulation Symposium, ANSS 2007, pp. 137–143. IEEE Computer Society, Washington, DC (2007)
18. Holcombe, M., Coakley, S., Smallwood, R.: A general framework for agent-based modelling of complex systems. In: Proceedings of the 2006 European Conf. on Complex Systems (2006)
19. Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K.: MASON: A New Multi-Agent Simulation Toolkit. Simulation 81(7), 517–527 (2005)
20. Márquez, C., César, E., Sorribes, J.: A load balancing schema for agent-based spmd applications. In: International Conf. on Parallel and Distributed Processing Techniques and Applications, PDPTA (accepted 2013)
21. North, M.J., Collier, N.T., Ozik, J., Tatara, E.R., Macal, C.M., Bragen, M., Sydelko, P.: Complex adaptive systems modeling with repast simphony. Complex Adaptive Systems Modeling 1(1), 1–26 (2013)
22. Oguara, T., Theodoropoulos, G., Logan, B., Lees, M., Dan, C.: Pdes-mas: A unifying framework for the distributed simulation of multi-agent systems. School of computer science research - University of Birmingham 6 (2007)
23. Rodin, V., Benzinou, A., Guillaud, A., Ballet, P., Harrouet, F., Tisseau, J., Le Bihan, J.: An immune oriented multi-agent system for biological image processing. Pattern Recognition 37(4), 631–645 (2004)
24. Scheutz, M., Schermerhorn, P., Connaughton, R., Dingler, A.: Swages-an extendable distributed experimentation system for large-scale agent-based alife simulations. Proceedings of Artificial Life X, 412–419 (2006)
25. Simon, H.A.: The architecture of complexity. Springer (1991)
26. Standish, R.K., Leow, R.: Ecolab: Agent based modeling for c++ programmers. arXiv preprint cs/0401026 (2004)
27. Suryanarayanan, V., Theodoropoulos, G., Lees, M.: Pdes-mas: Distributed simulation of multi-agent systems. Procedia Comp. Sc. 18, 671–681 (2013)
28. Tisue, S., Wilensky, U.: Netlogo: Design and implementation of a multi-agent modeling environment. In: Proceedings of Agent, vol. 2004, pp. 7–9 (2004)
29. Tobias, R., Hofmann, C.: Evaluation of free java-libraries for social-scientific agent based simulation. JASS 7(1) (2004)