Ubiquitous Computing and Distributed Agent-based Simulation

B.G.W. Craenen, G.K. Theodoropoulos
School of Computing
University of Birmingham
Birmingham, United Kingdom
b.g.w.craenen@cs.bham.ac.uk
g.k.theodoropoulos@cs.bham.ac.uk

***Abstract*—As much as ubiquitous computing systems are already claimed to exist in the real world, further development of these systems still pose challenges to computer science that are still quite beyond the state of the art. Two challenges stand out in particular: the complexity of next-generation ubiquitous computing systems, and their inherent scalability issues. This paper aims to establish that agent-based modelling provides a powerful tool in tackling these issues. As an example of a practical solution, readily available, this paper highlights the distributed agent-based simulation infrastructure PDES-MAS as particularly suited for the task. Using the PDES-MAS infrastructure, designers, developers, and builders of next-generation ubiquitous computing systems can, through an iterative agent-based simulation process, gain the required knowledge and information about these systems, without having precede to deployment of the system itself.**

***Keywords*-distributed simulation; agent-based systems; ubiquitous computing;**

I. INTRODUCTION

While computing devices have become ever more pervasive throughout every day life in modern society, the size and capabilities of the computing devices has changed dramatically as well. Computing devices, or computers, have become smaller, more capable and complex, and have become more and more interconnected. Computers have moved from large desktop models through smaller computers that can be carried onto a person's lap to mobile (phone) devices and computing pads that can be carried in a person's pocket. And the miniaturisation of computing devices is still ongoing with already promising work is being done on computing devices of a cubic centimeter and smaller.

In the post-desktop model of human-computer interaction, ubiquitous computing allows for information processing that is thoroughly integrated into everyday objects and activities. Someone using ubiquitous computing in the course of ordinary activities commonly engages many computing devices and systems simultaneously, and may not necessarily even be aware that they are doing so. A considerable advancement from the desktop paradigm already, ubiquitous computing is more formally defined as "machines that fit the human environment instead of forcing humans to enter theirs" [1].

The ubiquitous computing paradigm is also described as pervasive computing or ambient intelligence [2], where each term emphasizes slightly different aspects. Rather than propose a single definition for ubiquitous computing and these related terms, a taxonomy of properties for ubiquitous computing has been proposed, from which different kinds or flavours of ubiquitous systems and applications can be described [3].

At the core of all models of ubiquitous computing is a shared vision of small, inexpensive, robust networked processing devices, distributed at all scales throughout everyday life, turned generally to distinct common-place ends. Because this vision is still quite beyond the current state of the art, ubiquitous computing presents challenges across computer science: in systems design and engineering, in systems modelling, and in user interface design. Contemporary human-computer interaction models remain inappropriate and inadequate to the ubiquitous case and suggests that the interaction paradigm appropriate to a fully robust ubiquitous computing framework has yet to emerge.

This even though there is some recognition that in many ways humanity already lives in a ubiquitous computing world. In [4], Manuel Castells suggests that there is an ongoing shift from already-decentralised, stand-alone microcomputers and mainframes towards entirely pervasive computing. In his model, the example of the Internet is used as the start of a pervasive computing system with the logical progression from that paradigm, a system where networking logic becomes applicable in every realm of daily activity, in every location, and in every context. Castells envisages a system where billions of miniature, ubiquitous inter-communication devices will be spread worldwide, "like pigment in the wall paint", the latter echoing developments in speckled computing and smart dusts.

But that still leaves computer scientists the task of solving the problems in systems design and engineering, systems modelling, and user interface design. Leaving the latter for the human-computer interaction specialists, this paper will focus on the former two and focusses on discussing how agent-based modelling and simulation, and in particular distributed agent-based modelling and simulation, can play an important part.

And while the use of distributed agent-based modelling and simulation in ubiquitous computing in and of itself is not new, the application of these techniques is often topical and ad-hoc. More often than not it is applied to support solving the particular problem itself. And while this is obviously understandable, it ignores what the combination could contribute towards understand ubiquitous computing itself. In that respect, advances in the distributed multi-agent simulation field are there to be leveraged as well. And as an example, this paper will focus on the PDES-MAS system as a distributed multi-agent simulation system particularly well-suited for support ubiquitous computing in general.

The rest of the paper is then organised as follows. In section II we give a description of agent-based modelling and simulation with section III describing how it can be used to further ubiquitous computing in general. Section IV then describes the need for distribution with section V providing a means to do so with a description of the PDES-MAS platform. Finally in section VI we offer some conclusions and further discussion.

II. Agent-based modelling

An agent-based model (ABM) is a class of computational models for simulating the actions and interactions of autonomous agents in order to assess the effects on a system as a whole. Instead of ABM, sometimes the term multi-agent system (MAS), multi-agent simulation (also MAS), or individual-based models is used. The autonomous agents can describe both individual or collective entities such as organisations or groups within the system. ABMs simulate the simultaneous operations and interactions of multiple agents in an attempt to re-create and predict the appearance of complex phenomena. For this is a process of emergence from the micro to the macro level of the system. A key notion here is that simple behavioural rules generate complex behaviour, a principle extensively adopted in the modeling community. Individual agents are typically characterised as boundedly rational, presumed to be acting in what they perceive as their own interests.

Most computational modeling research describes systems in equilibrium, or as moving between equilibria. ABMs, however, uses simple rules to generate complex behaviour, usually resulting in far more complex and dynamic behaviours to examine. The three ideas central to ABM are social agents as objects, emergence, and complexity. Because ABMs consist of dynamically interacting rule-based agents, the systems within they interact, or of which they are part, can create real world-like complexity if the agents are: intelligent and purposeful as well as situated in space and time. The modeling process itself is best described as inductive, with the modeler making assumptions relevant to the situation and then watching phenomena emerge from the agents' interactions, where ABMs are generally seen to be complementing traditional analytic methods.

Where analytic methods focus on characterising the equilibria of systems, ABMs allow the possibility of generating those equilibria. It is this generative contribution to the analysis that may be the most mainstream of the potential benefits of ABMs. The ABMs' ability to explain the emergence of higher order patterns; their ability to identify those moments in time in which interventions have extreme consequences; and their ability to distinguish among types of path dependency; all make ABMs valuable analytic tools to use. Rather than focusing on stable states and equilibria, the ABM focuses on the system's robustness, i.e., the ways that complex systems adapt to internal and external pressures so as to maintain their functionalities. Harnessing that complexity requires consideration of the agents themselves: their diversity; connectedness; and their level of interactions are all worthy of consideration.

ABMs have been used since the mid-1990s to solve a variety of business and technology problems. Examples include supply chain optimisation and logistics, modelling of consumer behaviour, social network effect, distributed computing, and traffic congestion. In these and other applications, the system of interest is simulated by capturing the behaviour of individual agents and their interconnections. ABM tools can be used to test how changes in individual behaviours will affect the system's emerging overall behaviour. In addition, ABMs have been used to simulate information delivery in ambient assisted environments [5].

III. ABM and Ubiquitous Computing

Ubiquitous computing systems lend themselves well to be modelled by ABMs. The system itself consists of many computing devices, each with a clearly defined purpose, acting independently, and interacting with the other computing devices. The correlation between the computing devices that make up the ubiquitous computing system and autonomous and intelligent agents in an ABM is as such relatively straightforward. Using the ubiquitous computing system are humans, and their interactions with the system are also clearly defined and predominately independent from each other. They too can be encapsulated as agents in an ABM [6].

Modelling a ubiquitous computing system and it's users is, as such, relatively straightforward. But what is the benefit of doing so? The answer is that, the more complex a ubiquitous computing system becomes, the harder it becomes to design, develop and build. A ubiquitous computing system is almost by definition a dynamic and changing environment, and one where an equilibrium of provided services and functionalities with the requirements placed on the system is something to be aimed for in general. Of relevance here is that the system itself should be able to handle the dynamically changing circumstances as well, i.e., be robust.

Moreover, it is important to also capture and predict the emergence of those requirements, as they happen. To do so, modelling and simulation provides an obvious benefit with ABMs particularly suited. By encapsulating the ubiquitous computing system itself into agents, and having it interact with users, also encapsulated in agents, allows designers, and developers of the system to predict the use, requirements, and robustness of the system beforehand. In addition, the ABM allows these interested parties to analyse how these aspects of the ubiquitous computing systems came to be. Not only the emergence of the system itself can be studied, but emergence of behaviours when the system is altered or extended can be studied as well. If ubiquitous computing is the emergence of an embedded computing system, simulating the system through ABMs, and analysing the emergence behaviour, even before actually putting it into practice, can provide both information and behavioural patterns invaluable to the design and development of the system itself.

ABMs can provide further information on three aspects of ubiquitous computing valuable for ubiquitous computing: the interactions between the users and the system, as well as the system internally; the requirements placed on the system by the users; and the required functionality constraints of the individual parts of the system. As in an ABM both the users and the system itself are modelled as collections of agents, this boils down to investigating the interactions between the agents, and the behavioural rules of the agents themselves. This is, in essence, divide and conquer system development in action, with the ABM breaking down the complexity of the whole system and its users into relatively constituent parts governed by simple rules acting and interacting on and with each other.

The ABM then serves as a paradigm for reducing the complexity of the system as a whole into the complexity of its constituent parts and the complexity of the rules by which the controllers of these parts interact. The process, following the ABM paradigm, is in essence inductive, with a set of what-if scenarios providing information of the system without physical implementation or deployment being required. Although physical factors do certainly inform the ABM, concentrating primarily on the behaviour of the system allows for a level of abstraction, inductively diminished over time with increasingly complex models, that makes experimental prediction possible and relatively cheap. An ABM of a ubiquitous computing system allows the designer, developer, and builder of the system to predict and define: the interactions within the system; the requirements of each constituting part of the system; and the functionality, or rule-set, of (the controller of) each constituting part (for examples see [7], [8], for an alternate architecture using ABM for ubiquitous computing see [9]).

The idea of using ABM for ubiquitous computing is not entirely new however, and several research projects have made use of it. The FireGrid project (e.g. [10]) for example uses distributed and grid computing for emergency response applications relying on ABM techniques. But there is no unified approach to develop these sort of applications, all seem to be ad-hoc solutions for particular problems. And in general, advances in the distributed ABM field are available for application. In this respect we here like to focus on PDES-MAS, as a distributed agent-based modelling infrastructure well-suited to be used in support of agent-based modelling of ubiquitous computing systems. The use of a DSM systems is well suited to support the fluid nature of ubiquitous computing systems with respect to the location of the computing devices that form such an important part of it. The optimistic synchronisation technique complements the dynamic nature of ubiquitous computing systems equally well. With user interaction with those computing devices, and the interaction between the ubiquitous computing system devices so unpredictable, the PDES-MAS infrastructure and system provides a viable alternative for handling such circumstances.

IV. Distributed simulation

As much as ABM can provide a means for tackling the inherent complexity of ubiquitous computing systems, part of this complexity is a problem shared with ABM: scalability. Larger scale systems, be they ubiquitous computing systems, or ABMs in general, become more complex the larger they become. Ubiquitous computing systems, if one accepts the notion that they are already existent in society, are becoming larger, and the complexity of its constituent parts as well as its interactions has increased likewise. Without introducing increasing levels of abstraction into the model, ABMs of such ubiquitous computing systems will have to follow suit.

Part of this problem can be handled by using the inductive approach central to agent-based modelling. Modelling ubiquitous computing systems would start with a relatively small, highly abstract, low complexity ubiquitous computing system, to gradually, over various iterations, remove more and more layers of abstraction, introducing more and more constituent parts of the system. The process in itself will be valuable, and with proper validation at each step of the process, a close approximation of the entire system will be arrived at.

This leaves, however, technical limitations with the simulation itself. Building and refining an ABM is not only an inductive process, but a technical one as well. ABMs depend on an experimental simulation process, and the analysis of the results thereof. Validation of the model is then matching the behaviour of the ABM with the observed or expected behaviour of the system at each step in the process. Running those simulations however is possible only when enough computing resources are available to do so. With increasingly larger and complex ABMs and simulations, the computational resources available on one computing system will inevitably run out. The solution to this is to combine and effectively utilise the computational resources of several computing systems; in other words: Distributed Agent-based Simulation.

Distributed agent-based simulation, or Distributed Multi-Agent Systems (DMAS), focuses on distributing a ABM, or MAS, over multiple computer platforms. Various approaches, resulting in various simulation-engines, have been proposed [11]. Some simulation-engines focus on partitioning the simulation topology into several semi-autonomous regions, each distributed over the computing resources available. Others partition the shared state of the simulation using a Distributed Shared Memory (DSM) system, and distributing this over the computing resources available. These, and other approaches, have specific limitations and advantages, leading to a differentiation of the various approaches to the systems they are used to support. In this paper we use the PDES-MAS framework [12] as an example of a DSM of the later approach.

The shared state partitioning approach, using a DSM system, can be shortly described as follows. An ABM consists of a number of agents interacting with themselves or other agents. The environment in an ABM can then be described as either unchanging (static), or be encapsulated in yet other agents. The agents can be said to go through a sense-think-act cycle, where each agent first senses its environment and other agents, considers (thinks about) this input, and then acts on it. Most often ABMs of this type are event-based, where each action or interaction consists of a time-stamped event. Individual agent behaviour is determined during the think sub-cycle, usually according to a rule-set of varying complexity.

During the course of the simulation, called an experiment, each agent maintains a state. That is, a variable-set that includes all the information encapsulated in the agent. Determining the behaviour can require the agent to sense information encapsulated in other agents. State can be internal (private) and not shared or not visible to other agents, or external (public) and shared or visible to other agents. As such, agents can only sense the shared state of other agents.

The shared state partitioning approach focusing on distributing the shared state of agents. The internal state of an agent can not be sensed and need not be distributed. By distributing the agents themselves over the available computing resources, exceeding local computing resources is avoided. The DSM then makes available to all computing resources all shared state of all agents over the computing network. Distributing an ABM using the DSM approach then consists of distributing the agents (and the static part of the simulation) over the available computing resources such that the local limitations are not exceeded, while making available the shared state of all agents to all other agents on all the available computing resources. The focus of the DSM is then to provide a scalable method of getting access to the collective shared state of the agents in the simulation. Scalability is important because while the simulation can still only be run on with sufficient computing resources, a scalable system can grow when more computing resources are available and required.

V. PDES-MAS

In this paper we will use the PDES-MAS framework as an implementation of a DSM system for ABMs [12]. It implements a DSM structure where the shared state of the agents is represented by Shared State Variables (SSV). SSVs are data-structures that store the time-stamped history of values of a particular variable over time [13]. The collective state of the shared variables, encapsulated in SSV, can then be seen as similar to the ABM simulation's space-time memory [14], [15]. The shared variables stored in SSVs offer a natural representation of the simulation context when interactions between agents are described as interactions on these variables. Consequently we assume that agents alter the state of the ABM simulation by interacting with these SSVs in an event-based fashion.

The PDES-MAS framework then acts to make the space-time memory available across the different distributed computer platforms without exposing the exact way in which the memory is accessed or organised internally. Access to the SSVs is provided through read and write operations performed as events. These events allow the reading of a value stored in the SSV (ID-query), and writing a value in a certain SSV (ID-write) [16]. Reading of values over a range of SSVs satisfying a condition (associative memory) is an extension of the ID-query event, and is called a range-query. Range-queries are used to retrieve aggregate information during the sense-cycle, e.g., retrieving the location of all agents within viewing range [17].

Following the parallel distributed event simulation (PDES) paradigm, agents in the ABM are assigned to Logical Processes (LP), known as either Agent Logical Processes (ALP) that handle the agents themselves, or Communication Logical Processes (CLP) where SSVs are assigned. An ALP potentially models more than one agent, with multiple ALPs allowed concurrent access to the set of SSVs associated to the agents by connecting the ALPs to a tree-like network of CLPs. Conceptually, the ALP provides an access-node for the ABM simulation to the PDES-MAS framework. SSVs are then distributed among the CLPs in a scalable and balanced manner. Figure 1 shows a depiction of the PDES-MAS framework with 4 ALPs, 3 CLPs, and 4 SSVs.

Agents in a ABM are assumed to be intelligent with agents interacting with each other and their environment (partly also captured within agents). Because agents act independently and intelligently, based on sensed information, it is often hard to predict these interactions in advance, indeed discover them at all. As such, a defining characteristic of agents is their autonomy [18]. Moreover, how agents interact is often the primary goal of the simulation, particularly when considering ubiquitous computing. The sense-think-act cycle is then used to determine the behaviour translated

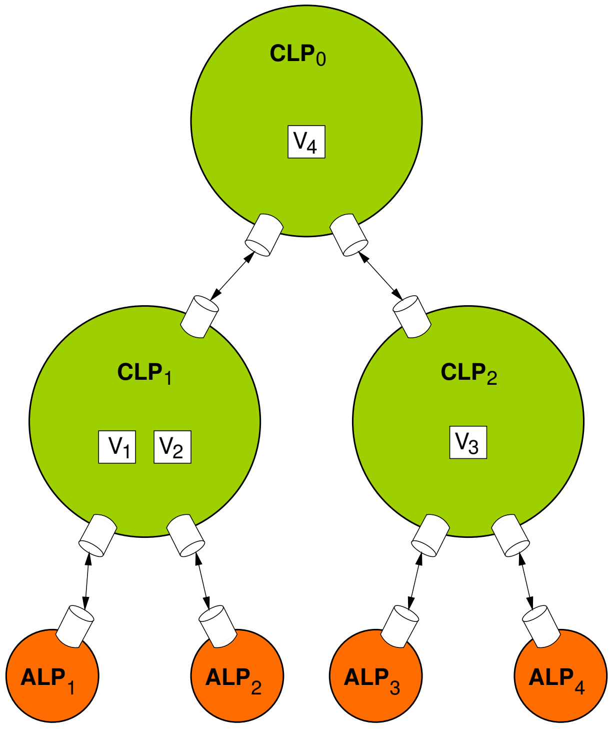

Figure 1. The PDES-MAS framework.

into actions with agents, over the life-time of the simulation, going through many of the cycles. Within the sense-cycle the PDES-MAS framework provides ID-query and Range-query events, while during the act-cycle it provides ID-write events. ALPs link to the leaf (parent) CLPs in the tree-like structure with ALPs, issuing requests to access SSVs through their parent CLPs. If the SSV is not assigned to the parent CLP, the access requests are passed along the tree until the SSV is located. The relevant data is the returned along the same route in reverse, back to the parent CLP, and from there to the ALP. Control messages internal to the PDES-MAS framework are conveyed through the tree in a similar way.

The PDES-MAS framework is used to provide simultaneous distributed access to the shared data of the ABM simulation and in this context maintaining data consistency is an important part of the functionality of the framework. Two main synchronisation mechanisms for maintaining data consistency can generally be recognised: conservative synchronisation, and optimistic synchronisation. Conservative synchronisation disallows conflicting access to the data by predicting when conflicting access will occur to then apply strict access-rules (pre-emptive locking). Optimistic synchronisation allows free access to the data at all times, but repairs data inconsistency afterwards through a roll-back mechanism [13].

Data consistency in PDES-MAS, and as such synchronising the events received from the ALP, is the responsibility of the CLP. The PDES-MAS framework uses optimistic synchronisation (see [19], [20], [21], [13], [22], [23], [24]). Each SSV is associated with a list of Write Periods (WP), each representing the values of the SSV at different times throughout the simulation. When a WP is invalidated by a straggler write, any agents associated to the ALPs that have read that WP with a subsequent time-stamp as the straggler write will be asked to roll-back to the time-stamp preceding the one of the straggler write. A rolled-back agent then resumes its time-progressing from before the data was invalidated, i.e., any inconsistenties in the data will be ignored and the data consistency itself will be repaired for that agent.

The CLP tree structure in the PDES-MAS framework is reconfigured dynamically and automatically, reflecting the interaction patterns of the agents in the overlaying ABM simulation based on the access patterns of the SSVs. SSVs accessed more frequently by an ALP are moved closer in the CLP tree structure to that ALP, i.e., the SSV is moved towards the parent leaf CLP of the ALP. The aim is to concurrently minimise the average number of hops required to access the SSV, as well as to reduce the load imbalance between the CLPs. In principle reconfiguration of the CLP tree structure can be achieved by creating or deleting CLPs; moving ALPs to different parent CLPs, or by migrating SSVs between CLPs. The PDES-MAS framework described in this paper implements a fixed binary tree-structure with leaf CLPs hosting a fixed and constant number of ALPs. Only SSVs are migrated through the tree to achieve redistribution [25].

VI. Conclusion

This paper discusses how distributed agent-based modelling and simulation can be a useful tool for designing, developing, and building ubiquitous computing. The spread and availability of computing resources and devices has increased dramatically in resources. The way these resources are made available, how these devices connect with each other, and how they interact with their human users has become ever more complex. Agent-based modelling and simulation can provide an invaluable tool to understanding and analysing this complexity, as well as providing a means for designing, developing, and building future (extensions to) ubiquitous computing systems.

But as ubiquitous computing becomes ever more pervasive and as ubiquitous computing systems increase in scale, so will the agent-based models of these systems if they are to provide analysis and information of sufficient fidelity. Increasingly large-scale agent-based models will inevitably out-grow the computing resources available on single computer platforms. For this problem, distributed multi-agent systems provide an answer. Considering the various requirements placed on distributed multi-agent systems by ubiquitous computing agent-based models this paper suggests the PDES-MAS framework as an already available and tested platform.

References

[1] J. York and P. Pendharkar, "Human-computer interaction issues for mobile computing in a variable work context," *International Journal of Human-Computer Studies*, vol. 60, pp. 771–797, 2004.

[2] U. Hansmann, *Pervasive Computing: The Mobile World*. Springer, 2003.

[3] S. Poslad, *Ubiquitous Computing Smart Devices, Smart Environments and Smart Interaction*. Wiley, 2009.

[4] M. Castells, *The Rise of the Network Society, The Information Age: Economy, Society, and Culture*. Blackwell, 1996, vol. 1.

[5] M. A. Niazi, "Self-organized customized content delivery architecture for ambient assisted environments," in *UPGRADE '08: Proceedings of the third international workshop on Use of P2P, grid, and agents for the development of content networks*, 2008, pp. 45–54.

[6] R. Ashri and M. Luck, "An agent construction model for ubiquitous computing devices," in *Agent-Oriented Software Engineering*, ser. Lecture Notes in Computer Science, J. Odell, P. Giorgini, and J. P. Müller, Eds. Springer Berlin - Heidelberg, 2005, vol. 3382, pp. 158–173.

[7] N. Sánchez-Pi, E. Mangina, J. Carbó, and J. Molina, "Multi-agent system (mas) applications in ambient intelligence (ami) environments," in *Trends in Practical Applications of Agents and Multiagent Systems*, ser. Advances in Soft Computing, Y. Demazeau, F. Dignum, J. Corchado, J. Bajo, R. Corchuelo, E. Corchado, F. Fernndez-Riverola, V. Julin, P. Pawlewski, and A. Campbell, Eds. Springer Berlin / Heidelberg, 2010, vol. 71, pp. 493–500.

[8] G. O'Hare and M. O'Grady, "Gulliver's genie: a multi-agent system for ubiquitous and intelligent content delivery," *Computer Communications*, vol. 26, no. 11, pp. 1177–1187, 2003.

[9] J. Soldatos, I. Pandis, K. Stamatis, L. Polymenakos, and J. L. Crowley, "Agent based middleware infrastructure for autonomous context-aware ubiquitous computing services," *Computer Communications*, vol. 30, no. 3, pp. 577–591, 2007.

[10] P. Massingberd-Mundy, "Firegrid an integrated emergency response system," in *14th International Conference on Automatic Fire Detection (AUBE'09)*, Duisburg, Germany, Sep 2009.

[11] G. Theodoropoulos, R. Minson, R. Ewald, and M. Lees, "Simulation engines for multi-agent systems," *Multi-Agent Systems: Simulation and Applications*, pp. 77–108, 2007.

[12] B. Logan and G. Theodoropoulos, "The distributed simulation of multi-agent systems," in *Proceedings of the IEEE*, vol. 89, Feb 2001, pp. 174–186.

[13] M. Lees, B. Logan, and G. Theodoropoulos, "Adaptive optimistic synchronisation for multi-agent simulation," in *Proceedings of the 17th European Simulation Multiconference (ESM 2003)*, D. Al-Dabass, Ed. Delft: Society for Modelling and Simulation International and Arbeitsgemeinschaft Simulation, Society for Modelling and Simulation International, 2003, pp. 77–82.

[14] K. Ghosh and R. M. Fujimoto, "Parallel discrete event simulation using space-time memory," in *Proceedings of the International Conference on Parallel Processing, Volume III, Algorithms & Applications*, 1991, pp. 201–208.

[15] H. Mehl and S. Hammes, "Shared variables in distributed simulation," in *Proceedings of the Seventh Workshop on Parallel and Distributed Simulation (PADS'93)*, 1993, pp. 68–75.

[16] D. Chen, R. Ewald, G. K. Theodoropoulos, R. Minson, T. Oguara, M. Lees, B. Logan, and A. M. Uhrmacher, "Data access in distributed simulations of multi-agent systems," *The Journal of Systems and Software*, vol. 81, pp. 2345–2360, May 2008.

[17] V. Suryanarayanan, R. Minson, and G. Theodoropoulos, "Synchronised range queries in distributed simulations of multi-agent systems," in *13th IEEE International Symposium on Distributed Simulation and Real Time Applications (DS-RT 2009)*, Singapore, Oct 2009.

[18] M. Wooldridge and N. R. Jennings, "Intelligent agents: Theory and practice," *Knowledge Engineering Review*, vol. 10, pp. 115–152, 1995.

[19] M. Lees, B. Logan, C. Dan, T. Oguara, and G. Theodoropoulos, "Decision-theoretic throttling for optimistic simulations of multi-agent systems," in *Proceedings of the Ninth IEEE International Symposium on Distributed Simulation and Real Time Applications (DS-RT 2005)*, A. Boukerche, S. J. Turner, D. Roberts, and G. Theodoropoulos, Eds. Montreal, Quebec, Canada: IEEE Press, Oct 2005, pp. 171–178.

[20] ——, "Analysing the performance of optimistic synchronisation algorithms in simulations of multi-agent systems," in *Proceedings of the 20th Workshop on Principles of Advanced and Distributed Simulation (PADS06)*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 37–44.

[21] ——, "Analysing probabilistically constrained optimism," in *Proceedings of the 10th IEEE International Symposium on Distributed Simulation and Real Time Applications (DS-RT 2006)*, E. Alba, S. J. Turner, D. Roberts, and S. J. Taylor, Eds. Malaga, Spain: IEEE Press, Oct 2006, pp. 201–208.

[22] M. Lees, B. Logan, and G. Theodoropoulos, "Time windows in multi-agent distributed simulation," in *Proceedings of the 5th EUROSIM Congress on Modelling and Simulation (EuroSim04)*, Paris, Sep 2004.

[23] ——, "Using access patterns to analyze the performance of optimistic synchronization algorithms in simulations of mas," *Transactions of the Society for Computer Simulation International*, vol. 84, pp. 481–492, 2008.

[24] ——, “Analysing probabilistically constrained optimism,” *Concurrency and Computation: Practice and Experience Journal, special issue on Distributed Simulation, Virtual Environments and Real Time Applications*, vol. 21, pp. 1467–1482, Aug 2009.

[25] T. Oguara, D. Chen, G. Theodoropoulos, M. Lees, and B. Logan, “An adaptive load management mechanism for distributed simulation of multi-agent systems,” in *The 9-th IEEE International Symposium on Distributed Simulation and Real Time Applications (DSRT 2005)*, Montreal, Oc. Canada, Oct 2005, pp. 179–186.