Event-Driven Multi-agent Simulation

Ruth Meyer(✉)

Centre for Policy Modelling, Manchester Metropolitan University, Oxford Road, Manchester M15 6BH, UK
r.meyer@mmu.ac.uk

Abstract. Most agent-based models today apply a time-driven approach, i.e. simulation time is advanced in equidistant steps. This time advance method is considerably easier to implement than the more flexible and efficient event-driven approach.

Applying the event-driven approach requires that (a) the durations for agent and environment actions are determined before they terminate, (b) each agent is able to instantly react to changes in its environment, and (c) the update of the state of the environment can be kept efficient despite updating agents asynchronously.

The simulation toolkit FAMOS fulfils these requirements, extending an existing discrete-event simulator. The toolkit also supports a flexible representation of space and the movement of agents in that space. These are areas where existing toolkits for agent-based modelling show shortcomings, despite the fact that a majority of multi-agent models explicitly model space and allow for mobile agents.

Keywords: Event-driven time advance · Discrete event simulation · Agent-based simulation · Spatially explicit agent-based model

1 Introduction

The last ten years have seen a surge in agent-based simulation models. Several disciplines have since adopted the multi-agent approach as a new paradigm for undertaking research; amongst them the social sciences [11], economics [36], geography [15] and ecology [14]. Across disciplines, agent-based models are applied to investigate complex systems [10]. One of the main reasons for this expansion is the availability of software toolkits, which support agent-based modelling and simulation well enough to make the approach attractive for domain experts in a variety of application areas [33].

Agent-based simulation views the system to be modelled as a multi-agent system, i.e. as consisting of autonomous agents interacting in and with an environment. To build an agent-based model a modeller has to specify both the structure of the model and its dynamic behaviour over time. Simulation toolkits therefore need to provide constructs to implement a set of agents, their relationships, which influence their interactions, and their joint environment, which may be spatially explicit and contain dynamic processes in addition to the agents.

F. Grimaldo and E. Norling (Eds.): MABS 2014, LNAI 9002, pp. 3–16, 2015.
DOI: 10.1007/978-3-319-14627-0_1

For the description of the dynamic behaviour the actions of agents and, if necessary, the environment have to be related to simulation time. There are several ways to do this. Most widely used is the time-driven approach [26], which divides simulation time into regular intervals (time steps) and updates all agents synchronously at every time step. The modeller needs to identify the agents' actions and specify the order in which they are to be executed at each time step (see e.g. [14], p. 111). Another approach is the event-driven time advance, which combines irregular time intervals with an asynchronous update of agents (see Sect. 2).

The predominance of the time-driven approach can be explained from both its ease of implementation with regard to the simulation infrastructure – no dedicated scheduler and event list are necessary – and its simplification of modelling the agents' behaviour, since all actions implicitly obtain a duration ($\leq \Delta t$, the length of the fixed time step) and a modeller only has to specify the order of actions occurring during the same time step. Tutorials for existing software toolkits and introductory textbooks consolidate the use of the time-driven approach in their example models (see e.g. [13,23,25,27,28]).

Moreover, an analysis of existing toolkits for agent-based simulation shows that toolkits that allow for an event-driven time advance like Repast [29], Swarm [27] or James II [16] offer little or no support for the implementation of agents and environment, whereas toolkits with such functionality like NetLogo [41], SeSam [18] or Jason [7] tend to constrain model execution to the time-driven approach. This indicates that "there is a gap in the current design space for a toolkit which both provides/prescribes some structure for implementing agents but also provides a full Discrete Event scheduling implementation for the model's execution" ([37], p. 85).

Thus, agent-based models using an event-driven approach are still rare (examples are [38] and [22]), even though this approach has been tried and tested for decades in traditional discrete-event simulation (see e.g. [2,21]).

In the following, we first discuss the advantages of an event-driven time advance (Sect. 2) before introducing a toolkit for multi-agent simulation that applies this approach (Sect. 3). Two example models, a re-implementation of Schelling's famous segregation model (Sect. 4) and the model of a city courier service (Sect. 5), demonstrate key features of this toolkit. The paper finishes with a discussion (Sect. 6) and conclusion (Sect. 7).

2 The Case for Event-Driven Time Advance

In the event-driven approach simulation time advances from one event to the next. Each event models a change in the system, e.g. an agent receiving a message or arriving at a particular location in space. The intervals between events can be of any length as they are determined by the processes occurring in the system. The state of the system is assumed constant between events.

There are several reasons to adopt an event-driven time advance in agent-based models. Firstly, this approach is more efficient than the time-driven approach, since it regards only those points in time when changes actually occur and skips inactive phases of the system. In addition, only entities affected by the current event have to be updated. Secondly, it is more accurate as it allows events to occur at the correct time whereas the time-driven approach "collects" all events occurring within one time step and treats them as if they are happening at the same time, i.e. the end of the fixed time interval. Thirdly, this leads to it being more flexible, since it can accommodate a heterogeneous discretisation of time (e.g. periods of time with many events happening close together interspersed with periods where only a few events occur) as well as agents operating on different time scales.

While for many systems a time-driven simulation is well suited or at least adequate, other systems require a correct mapping of events to points in simulation time. Examples can be found in different application domains, from competition over particular habitats in ecology (e.g. [14], p. 112) to chemical reactions ([3], p. 26ff) to financial markets ([6,8,17]). A time-driven approach would either falsify results due to treating consecutive events as happening simultaneously if the fixed time interval Δt is too big, or would be very inefficient because Δt has to be chosen as the smallest interval between two events. The latter introduces the additional problem that this interval is often not known beforehand so that several simulation runs are required to determine it. Since a time-driven approach can easily be mapped to an event-driven approach, the event-driven time advance allows for the simulation of a more comprehensive class of models.

In addition, it can be argued that the fixed clock rate and synchronous agent update of the time-driven approach defies the concept of autonomy that underlies the definition of agents in multi-agent systems: "Forcing all agents of the MAS to act in lock step does not fit with autonomy of agents" ([40], p. 177). An event-driven approach is more suitable as it does not a priori impose synchronisation on agents ([4], p. 269). Social processes in particular are rarely synchronous, which is why it is equally rarely appropriate to model them by updating agents synchronously once per time step ([1], p. 41).

3 A Framework for Event-Driven Multi-agent Simulations

The Framework for Agent-oriented Modelling and Simulation (FAMOS[1]) combines multi-agent simulation with an event-driven time advance and an explicit representation of space. A comparison with existing agent-based simulation toolkits poses the question how necessary a new toolkit could be. Several ABMS toolkits contain a discrete-event scheduler within their features; the most widely used are Swarm [27], Mason [24] and Repast [29] or Repast Simphony [28]. Using the event-driven approach in agent-based models is therefore possible. We would like to argue though, that it is not sufficient to provide the necessary simulation infrastructure. Its use also has to be adequately supported and here current simulation toolkits are lacking.

On the one hand, their documentation focusses on how to implement time-driven models (see e.g. [27], p. 3; [23], p. 16f; [28], p. 18ff). On the other hand,

[1] Available at http://famos.sourceforge.net.

the constructs they provide to model agent behaviour, environment including space, and interactions between agents and environment have not been adapted to the requirements of an event-driven time advance. These requirements include

1. determining the duration of actions so that the respective termination event can be scheduled in time;
2. enabling each agent to instantly react to changes in its environment, even though it is technically passive during time-consuming phases;
3. ensuring an efficient update of the environment state (which may include the current positions of mobile agents) despite updating agents asynchronously.

The toolkit FAMOS presented in this paper addresses these requirements. It supports event-driven multi-agent simulation appropriately by not only providing the necessary simulation infrastructure but also dedicated constructs for modelling agent behaviour and spatially explicit environments that have been adapted for use with an event-driven time advance.

3.1 Modelling Time

To avoid re-inventing the wheel, FAMOS builds on an existing discrete-event simulation framework and extends it with multi-agent simulation features. DESMO-J[2] ([31], ch. 10) is an open-source software development framework for discrete-event simulation and supports two of the classical world views: event scheduling and process interaction. In addition to directly re-using DESMO-J's simulation infrastructure (scheduler, event list, simulation clock), agent-based models in FAMOS can use entities and event routines or simulation processes to model any dynamic behaviour of the environment that is not related to agents, e.g. renewable resources.

The requirements imposed by the event-driven time advance are addressed as follows:

1. The durations of model-independent actions like sensing the environment, manipulation of objects in the environment, sending messages or moving in space, which FAMOS provides, are determined automatically. In the current version of the framework, all actions are assumed to be instantaneous except for movement. The duration of a move is calculated from the current speed of the agent and the distance covered, which in turn is calculated by the space model. For model-specific actions that are not provided by FAMOS a modeller may use the stochastic distributions of the underlying simulation framework DESMO-J to determine durations.
2. To guarantee that each agent can instantly react to changes in its environment FAMOS lets the environment send all agents that are affected by the change a special notification signal, i.e. those agents in whose area of perception (defined by an agent-specific sensor range) the change occurred. This signal causes an agent to automatically be scheduled for re-activation so that it can interrupt its current, simulation-time consuming action and decide itself if and how it wants to react to the change.

[2] Available at http://desmoj.sourceforge.net.

3. Discrete-event simulation links all state changes to instantaneous events with no changes happening between events. Time-consuming actions have to be mapped to a series of events (at least start and end); their effect usually happens at the end event. In FAMOS this only poses a problem for the movement of agents as the current positions of mobile agents may become too inexact when they are updated at the end of a longer movement process. To be able to keep positions of mobile agents exact enough while avoiding an update of the environment every time before an agent might access it, longer movements across several cells or nodes are divided into small steps that are regarded as atomic transactions. An atomic step is defined as the movement between adjacent cells in a grid or adjacent nodes in a graph. The change of position is visible in the environment whenever an agent has crossed the border between two grid cells or has passed the first half of the edge between two nodes. Its event time is calculated automatically. The moving agent is re-activated after each step by receiving a notification from the environment, which enables it to review its situation and adapt its movement accordingly if need be.

This adaptation of movement to the requirements of the event-driven approach is encapsulated in a particular `Movement` component. Its method `move()` provides agents with the ability to move in space by automatically calculating the duration of the move and scheduling a respective re-activation event. The component can be adapted to model-specific needs by choosing or implementing a particular movement strategy, which is in charge of determining the next position to move to. Pre-defined strategies are walking randomly (`RandomWalk`), following a gradient (`GradientTrace`), moving in a given direction (`MoveInDirection`) and following a planned route (`MoveAlongPath`).

This support of an agent's movement exceeds the functionality of comparable simulation toolkits, where a modeller has to combine primitive commands (delete at current position, add at new position, update agent coordinates) to achieve a change of position ([32], p. 614).

3.2 Modelling Space

The flexible, discrete space representation is another of FAMOS's fortes. It combines the prevalent grids of agent-based models with directed graphs by using the fact that each tessellation possesses a dual graph. This relation between tessellations and graphs may be self-evident but is rarely used explicitly.[3] An exception are Voronoi diagrams, whose dual graphs are known as Delaunay Triangulations (see e.g. [5]). While the former can be used to solve nearest neighbour problems, the latter are applied e.g. in Geographic Information Systems as digital elevation models.

FAMOS's space representation regards space as made up of discrete space elements connected by neighbourhood relationships. These neighbourhood links influence an agent's perception and movement by determining which elements

[3] Another example is David O'Sullivan's combination of graphs with irregular cellular automata to model spatial processes in cities [30].

are accessible from the agent's current position. A directed graph is used to store the spatial structure by mapping space elements to nodes and neighbourhood relationships to directed edges. This mapping can be done automatically so that there is no additional effort required for the modeller. In the current version of FAMOS this is implemented for regular grids with square/rectangular cells (`RectangularGrid`) or hexagonal cells (`HexagonalGrid`) and irregular grids, which are defined as a set of points. From these, both the Voronoi diagram (`IrregularGrid2D`) and the dual graph are calculated.

Access to the space model is routed solely through the environment, which acts as a façade (according to the design pattern of the same name [12]) providing agents with an interface not only to the space but also to the communication infrastructure and organisational groups.

3.3 Modelling Agents

FAMOS abstracts from a particular agent architecture and adopts a modular approach. An agent possesses a number of abilities (communication, access to the environment, movement), which may be extended as needed, and an interchangeable behaviour component. In adaptation to the event-driven time advance, each agent has its own internal "event" list, which stores external signals (notifications from the environment, messages from other agents) and internal signals (generated by the agent itself) in chronological order. The agent automatically schedules itself for re-activation for the time of the most imminent next signal. On re-activation all imminent signals are passed to the behaviour component for processing.

At the moment four such components are implemented, offering different methods to model an agent's behaviour. The simplest is a variant of event-oriented modelling (`SimpleBehaviour`), in which the agent's reaction to signals can be specified by implementing the `process()` method. Proactive behaviour can be achieved by scheduling internal signals for certain points in time with the agent-internal methods `scheduleIn()` or `scheduleAt()`. This component is ideally suited to model large populations of reactive agents and is used in the example model described in the next section. The component implementing a variant of the process interaction world view (`ProcessBehaviour`) is particularly suited to model proactive behaviour, which is only occasionally interrupted by events. The rule-based component (`RuleEngine`) integrates the rule engine Jess[4] to allow for declarative behaviour modelling. The most comprehensive component (`StateMachine`) facilitates the use of state diagrams to model an agent's behaviour. These can be specified using a graphical editor and then automatically parsed into executable code.

4 Example 1: Schelling's Segregation Model

To test and demonstrate some of FAMOS's key features the well-known segregation model described by Thomas Schelling [34] was chosen to be re-implemented.

[4] http://www.jessrules.com/jess.

Though simple, this agent-based model nevertheless allows us to showcase both the advantages of the event-driven approach and the flexible space model.

In the model two types of agents live on a square grid neighbourhood. Each agent is content with its position if at least 3 of its 8 neighbours are of the same type (tolerance threshold 0.375). If this is not the case, it will move to a free position on the grid. While Schelling originally defined the new position to be the closest free cell where the agent would be content, in computer implementations of the model this is usually replaced by choosing a random free cell.

The version implemented in FAMOS follows this approach. Since the agents solely react on the state of their local environment their behaviour can easily be modelled with FAMOS's simplest behaviour component, the event-oriented `SimpleBehaviour`. Here, the modeller has to specify an agent's reaction to incoming signals (events). In the case of the segregation model, these are notifications from the environment whenever a change occurred within the agent's sensor range, i.e. another agent moved in or out of the neighbourhood, or the agent itself arrived at a new position. An agent's reaction consists of checking if it is still content with its position and – if that is not the case – moving to a randomly chosen new position.

The action of choosing a position and moving there is delegated to framework classes. FAMOS provides several movement strategies (see Sect. 3.2) that can either be used as is or extended by a modeller to adapt to their purposes. To enable agents to pick a random new position anywhere on the grid instead of just in their direct neighbourhood the existing `RandomWalk` strategy was sub-classed to include all free cells into the selection process. This new `SegregationStrategy` class was then plugged into one of the standard movement components of FAMOS. Since the model abstracts from the duration of the actual movement in that a move to a neighbouring cell is treated the same as a "jump" to the other end of the grid, the `ConstantTimeMovement` provides the right functionality here. Inside the behaviour specification, the modeller now just needs to call the `move()` method to make an agent select and then move to a new position.

Figure 1 shows the results of simulation runs with each of the three different grid spaces FAMOS provides. The left column pictures the situation at the start of a run, with the agents randomly scattered across the space, whereas the right column contains the situation at the end of the respective run. The runs differ only in the chosen representation of space, the other parameters have been kept the same across all runs. Each grid consists of 400 cells and is populated with 140 blue and 140 green agents, whose tolerance threshold is set to the original value of 0.375. The duration for a move to a new position is set to 1.0 units of simulation time.

5 Example 2: City Courier Service Model

A more complex example, which demonstrates FAMOS's comprehensive support for the movement of agents in space and its adaptation to the event-driven time

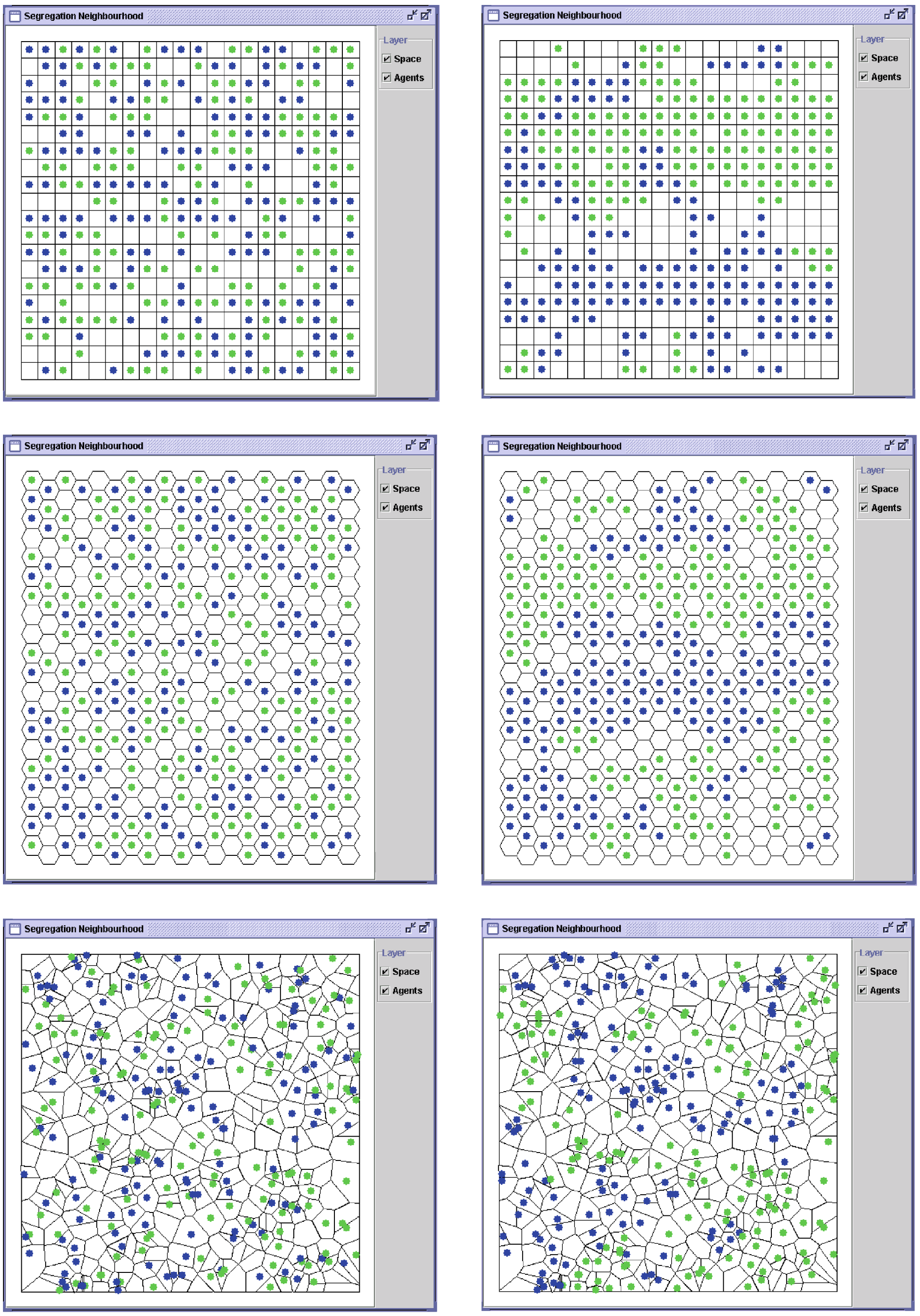


Fig. 1. Screenshots of the segregation model in FAMOS from simulation runs with the three different grids `RectangularGrid` (top), `HexagonalGrid` and `IrregularGrid2D` (bottom). The left column shows the situation at the start, the right column at the end of a simulation run.

advance, is the model of a city courier service [19,20]. The courier service in question consists of a fleet of bike and car couriers, who deliver orders throughout the city. They decide themselves which orders to take and plan their own route. A central office receives orders from clients and passes them on to the couriers via radio using a variant of the contract net protocol [35], which gives idle couriers priority and allows for special requests of clients.

The courier service system is naturally driven by events: Arrival, placement, pick up and delivery of orders as well as start and end of work of couriers. A model needs to map the occurrence of these events in real time correctly to simulated time to avoid falsifying results. The model also needs an explicit representation of space since the current positions of the couriers influence their decision making and thus the system dynamics.

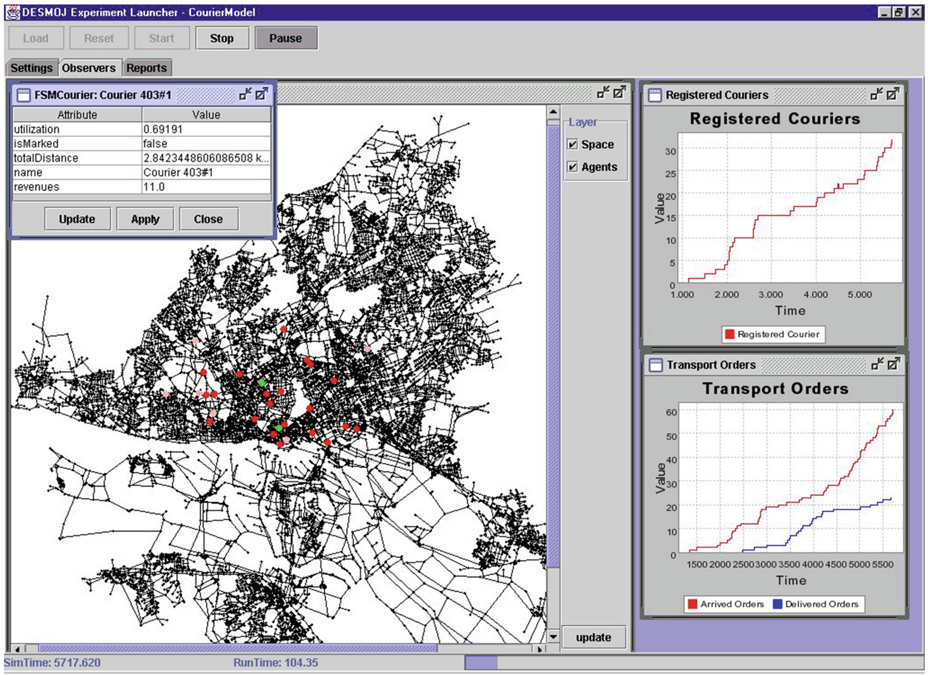
Using FAMOS such a model can be implemented with relatively little effort. The main task for a modeller is to specify the behaviour of the agents (office and couriers), while the agents' environment (space, communication and organisation structures) can be realised with the predefined components of FAMOS. Since FAMOS uses a directed graph as the underlying space model, the detailed road network of the city of Hamburg, consisting of more than 17,000 nodes and 48,000 edges, can be represented without problems (see Fig. 2). Edge attributes modelling road types influence speed and route choice of couriers in relation to their vehicle (bike or car). FAMOS's movement component is parameterised with a model-specific rating function, which determines the duration of a move along an edge depending on the courier's vehicle and the edge's attributes. This is the only model-specific adaptation of the framework's black box classes necessary for the courier model.

The behaviour of office and couriers is complex enough to warrant modelling on a higher level than the simple reactive or proactive components `SimpleBehaviour` and `ProcessBehaviour` provide. The fact that both individual behaviour and interaction of office and couriers are mainly controlled by events like arrival or delivery of an order suggests the use of the `StateMachine` component. The implementation of executable state diagrams in FAMOS is compliant with the UML semantics and supports hierarchical states, orthogonal regions, inter-level transitions and internal transitions, i.e. reactions to events that do not involve a state change. This makes it possible to model the part of a courier's behaviour regarding communication with the office as independent from the aspect of order processing. Only the deliberative aspects of courier behaviour (deciding which orders to take on and in which order to process them) are not yet supported by components in FAMOS and have to be implemented directly in the underlying programming language Java.

Figure 2 shows the screenshot of a simulation run with the courier service model, using empirical data from existing courier service companies amounting to 200 couriers and 2200 orders per day.

6 Discussion

The segregation model shows typical features of a discrete event system and is therefore well suited for the event-driven multi-agent simulation that this paper



Attribute	Value
utilization	0.69191
isMarked	false
totalDistance	2.8423448606086508 k...
name	Courier 403#1
revenues	11.0

SimTime: 5717.620 RunTime: 104.35

Fig. 2. Screenshot of the courier service model in FAMOS. Courier agents are coloured according to vehicle (red: car, green: bike) and current load (light: idle, dark: at least one order) (Color figure online).

proposes. After the initialisation phase, where all agents have to check if they are satisfied with their position, another check is only required if the local state of the environment has changed. All changes in this model are due to either an agent moving away from a position or an agent arriving at a new position. With a constant duration for each move this results in a quasi-time-driven simulation – but without the need to synchronously update all agents at each time "step".

Since FAMOS has agents receive a signal whenever a change occurs within their area of perception or they arrive at a new position, an event-driven control can be consistently implemented for the segregation model. A modeller just has to specify the appropriate reaction to this signal – checking if the agent is still content and possibly moving to a new position – in the agent's behaviour description.

The flexible space model allows to simulate the segregation model with different grids without having to adapt the description of the agent's behaviour. This is due to the explicit representation of locations as `Position` objects, which results in providing a layer of abstraction between the agents and the space model. In addition, using a directed graph as the underlying space representation allows for the uniform application of all movement strategies to different space models. Thus, the agents' access to their spatial environment is independent of a particular space model.

The courier service model is an example of a much more complex system that is also driven by events: The arrival of an order at the central office triggers the allocation process whose successful termination – awarding of an order to a courier – in turn triggers the pick-up and delivery process. Event-driven multi-agent simulation as proposed in this paper allows for a natural modelling of this system.

The empirical data available for the courier service model includes exact times for external events, i.e. those events that affect the system boundaries: arrival of orders and start and end of work of couriers. Depending on these events, the times of all other events arise from the simulation of the system. An event-driven time advance makes it possible to maintain the accuracy set by the empirical data – given that the duration of all relevant activities can be determined exactly enough. The courier service model applies the assumption that uniform, state-independent activities like the order-related actions of the office (receiving an order, announcing an order via radio, awarding an order to a courier) on average always take the same time; thus they are assigned a constant duration.

In contrast, the travel times of couriers and their response time to order announcements depend on the current state of the system and therefore have to be determined during the simulation. While the calculation of travel times can simply be delegated to the framework FAMOS due to its high-level support of movement in space, the determination of how quickly a courier responds to an order announcement via radio has to be done without framework support. In the model, it is approximated by a courier's interest in the order, which is calculated from the potential profit and the courier's current workload. The interest is assumed to be inversely proportional to the courier's reaction time, i.e. the higher the interest, the sooner the courier will offer to take the order.

7 Conclusion and Outlook

Systems that are inherently driven by events can be found in such diverse application domains as ecology (e.g. [14], p. 112), chemistry (e.g. [3], p. 26ff), financial markets (e.g. [6]) and the courier service described in Sect. 5. If correct timing of events is important for a system's behaviour event-driven time advance is necessary to adequately model such systems. The toolkit FAMOS and its application in the segregation model and the courier service model demonstrate that event-driven multi-agent simulation is feasible when appropriately supported. This means that not only are the technical requirements met by providing a suitable simulation infrastructure, but also that support is offered at the level of the agents and the environment. Particular focus lay on developing a flexible representation of space and comprehensively supporting movement in space because the majority of agent-based models use an explicit space model with mobile agents [9] and the movement needs to be adapted to the event-driven approach.

Many aspects of FAMOS have so far been implemented only provisionally. This pertains in particular to the support of data analysis and validation, for which only the minimum requirements like recording simulation output data and providing visualisations during a simulation run are met. Further components to specify cognitive or deliberate agent behaviour are desirable. In addition, combining several, task-specific behaviour components, which could be exchanged at run-time, would allow for adaptive agents.

References

1. Axtell, R.: Effects of interaction topology and activation regime in several multi-agent systems. In: Moss, S., Davidsson, P. (eds.) MABS 2000. LNCS (LNAI), vol. 1979, pp. 33–48. Springer, Heidelberg (2001)
2. Banks, J., Carson, J.S., Nelson, B.L., Nicol, D.: Discrete-Event System Simulation, 3rd edn. Prentice Hall, Upper Saddle River (2000)
3. Barnes, D.J., Chu, D.: Introduction to Modeling for Biosciences. Springer, London (2010)
4. Baveco, J.M., Lingeman, R.: An object-oriented tool for individual-oriented simulation: host-parasitoid system application. Ecol. Model. **61**, 267–286 (1992)
5. de Berg, M., van Kreveld, M., Overmars, M., Schwarzkopf, O.: Computational Geometry: Algorithms and Applications. Springer, Heidelberg (1997)
6. Boer, K., Kaymak, U., Spiering, J.: From discrete-time models to continuous-time, asynchronous models of financial markets. Comput. Intell. **23**(2), 142–161 (2007)
7. Bordini, R.H., Hübner, J.F.: Agent-based simulation using BDI programming in Jason. In: Uhrmacher and Weyns [39], pp. 451–476
8. Daniel, G.: Asynchronous Simulations of a Limit Order Book. Dissertation, University of Manchester, Faculty of Science and Engineering (2006)
9. Davidsson, P., Holmgren, J., Kyhlbäck, H., Mengistu, D., Persson, M.: Applications of agent based simulation. In: Antunes, L., Takadama, K. (eds.) MABS 2006. LNCS (LNAI), vol. 4442, pp. 15–27. Springer, Heidelberg (2007)
10. Edmonds, B., Meyer, R. (eds.): Simulating Social Complexity: A Handbook. Understanding Complex Systems. Springer, Berlin (2013)
11. Epstein, J.M.: Generative Social Science: Studies in Agent-Based Computational Modeling. Princeton University Press, Princeton (2007)
12. Gamma, E., Helm, R., Johnson, R.E., Vlissides, J.M.: Design Patterns - Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading (1994)
13. Gilbert, N., Troitzsch, K.G.: Simulation for the Social Scientist, 2nd edn. Open University Press, Maidenhead (2005)
14. Grimm, V., Railsback, S.F.: Individual-Based Modeling and Ecology. Princeton series in theoretical and computational biology. Princeton University Press, Princeton (2005)
15. Heppenstall, A.J., Crooks, A.T., See, L.M., Batty, M. (eds.): Agent-Based Models of Geographical Systems. Springer, Dordrecht (2012)
16. Himmelspach, J., Uhrmacher, A.M.: Plug'n simulate. In: Proceedings of the 40th Annual Simulation Symposium (ANSS-40 2007), Norfolk, VA, 26–28 March 2007, pp. 137–143. IEEE Computer Society (2007)
17. Jacobs, B.I., Levy, K.N., Markovitz, H.M.: Financial market simulation in the 21st century. J. Portfolio Manage. (30th Anniversary Issue) **30**, 142–151 (2004)

18. Klügl, F., Herrler, R., Fehler, M.: Sesam: implementation of agent-based simulation using visual programming. In: Nakashima, H., Wellman, M.P., Weiss, G., Stone, P. (eds.) Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), Hakodate, Japan, 8–12 May 2006, pp. 1439–1440. ACM (2006)
19. Knaak, N., Meyer, R., Page, B.: Agent-based simulation of sustainable logistic strategies for large city courier services. In: Proceedings of EnviroInfo 2003, 17th International Conference Informatics for Environmental Protection, Cottbus, pp. 318–325, September 2003
20. Knaak, N., Meyer, R., Page, B.: Logistic strategies for sustainable city courier services - an agent-based simulation approach. In: Proceedings of HMS 2004, 8th International Workshop on Harbour, Maritime & Multimodal Logistics Modelling and Simulation, Rio de Janeiro, September 2004
21. Law, A.M., Kelton, W.D.: Simulation Modeling and Analysis, 3rd edn. McGraw-Hill, Boston (2000)
22. Lawson, B.G., Park, S.: Asynchronous time evolution in an artificial society model. J. Artif. Soc. Soc. Simul. **3**(1) (2000). http://jasss.soc.surrey.ac.uk/3/1/2.html
23. Luke, S.: Multiagent simulation and the MASON library. Manual version 17, Department of Computer Science, George Mason University, Fairfax, VA, May 2013, http://cs.gmu.edu/~eclab/projects/mason/manual.pdf
24. Luke, S., Cioffi-Revilla, C., Panait, L., Sullivan, K., Balan, G.: MASON: a multi-agent simulation environment. Simulation **82**(7), 517–527 (2005)
25. Macal, C.M., North, M.J.: Agent-based modeling and simulation: Abms examples. In: Mason, S.J., Hill, R.R., Mönch, L., Rose, O., Jefferson, T., Fowler, J.W. (eds.) Proceedings of the 2008 Winter Simulation Conference, pp. 101–112 (2008)
26. Michel, F., Ferber, J., Drogoul, A.: Multi-agent systems and simulation: a survey from the agents community's perspective. In: Uhrmacher and Weyns [39], pp. 3–52
27. Minar, N., Burkhart, R., Langton, C.G., Askenazi, M.: The swarm simulation system: a toolkit for building multi-agent simulations. Working Paper 96-06-042, Santa Fe Institute (1996), http://www.santafe.edu/media/workingpapers/96-06-042.pdf
28. 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 Adapt. Syst. Model. **1**, 3 (2013). http://www.casmodeling.com/content/1/1/3
29. North, M.J., Collier, N.T., Vos, J.R.: Experiences creating three implementations of the Repast agent modeling toolkit. ACM Trans. Model. Comput. Simul. **16**(1), 1–25 (2006)
30. O'Sullivan, D.: Graph-based Cellular Automaton Models of Urban Spatial Processes. Dissertation, Centre of Advanced Spatial Analysis, University of London (2000)
31. Page, B., Kreutzer, W.: The Java Simulation Handbook: Simulating Discrete Event Systems with UML and Java. Shaker, Aachen (2005)
32. Railsback, S.F., Lytinen, S.L., Jackson, S.K.: Agent-based simulation platforms: review and development recommendations. Simulation **82**(9), 609–623 (2006)
33. Samuelson, D.A., Macal, C.M.: Agent-based simulation comes of age. Oper. Res./Manage. Sci. Today **33**(4), 34 (2006)
34. Schelling, T.C.: Micromotives and Macrobehavior. Norton, New York (1978)
35. Smith, R.G.: The contract net protocol: high level communication and control in a distributed problem solver. IEEE Trans. Comput. **C–29**(12), 1104–1113 (1980)
36. Tesfatsion, L.: Agent-based computational economics: growing economies from the bottom up. Artif. Life **8**(1), 55–82 (2002)

37. Theodoropoulos, G., Minson, R., Ewald, R., Lees, M.: Simulation engines for multi-agent systems. In: Uhrmacher and Weyns [39], pp. 77–108
38. Troitzsch, K.: A multi-agent model of bilingualism in a small population. In: Coelho, H., Espinasse, B. (eds.) 5th Workshop on Agent-Based Simulation, pp. 38–43. SCS Publishing House, Erlangen (2004)
39. Uhrmacher, A.M., Weyns, D. (eds.): Multi-Agent Systems: Simulation and Applications. CRC Press/Taylor and Francis, Boca Raton (2009)
40. Weyns, D., Holvoet, T.: Model for situated multi-agent-systems with regional synchronization. In: Jardim-Goncalves, R., Cha, J., Steiger-Garcao, A. (eds.) Enhanced Interoperable Systems: Proceedings of the 10th International Conference on Concurrent Engineering (ISPE CE 2003), Madeira, Portugal, 26–30 July, pp. 177–188 (2003)
41. Wilensky, U.: Netlogo. Center for Connected Learning and Computer-Based Modeling, Northwestern University, Evanston (1999). http://ccl.northwestern.edu/netlogo/