ABSTRACT

Agent-Based Modelling and Simulation (ABM/S) is still struggling to become one of the main stream simulation methods in Operational Research (OR) and Management Science (MS), despite its generally accepted usefulness when it comes to representing human behaviour in human-centric systems. In other fields, as for example Business Studies, Economics, and Social Science, it is flourishing. One of the technical differences between ABM/S and the well-established OR/MS simulation methods System Dynamics Simulation (SDS) and Discrete Event Simulation (DES) is that ABM/S traditionally uses an equation based modelling approach while SDS and DES use a graphical notation for the model description. We believe that having a graphical notation for ABM/S would help establish it in OR/MS. The Unified Modelling Language (UML) is a graphical notation commonly used in software engineering for the purpose of software design. Use case and state machine diagrams, which are part of the UML notation seem to lend themselves particularly well to ABM/S. In this paper we introduce UML to the OR/MS community. First we explain step-by-step how to use UML for developing ABM/S models. Then we demonstrate the application of this graphical notation by presenting two conceptual models we built for real world OR/MS case studies.

Keywords: Operational Research, Management Science, agent-based modelling, agent-based simulation, UML, model notation, conceptual modelling, use case diagram, state machine diagram, state chart

1 INTRODUCTION

The topic of the panel discussion at the 2010 Operational Research Society Simulation Workshop was the relevance of Agent-Based Modelling and Simulation (ABM/S) in Operational Research (OR) and Management Science (MS)\(^1\). The discussion highlighted that representing human behaviour within operations and service systems with the help of ABM/S was still in its infancy in OR (Siebers et al 2010). We were interested to see if anything had changed over the last 4 years. A quick search in the International Abstracts in Operations Research (IAOR) database for two 4 year periods using keywords related to the topic of OR simulation\(^2\) shows that the reported use of ABM/S has nearly doubled within the last 4 years (Table 1). But we need to interpret this result with care. Spot checks uncovered that most of these papers relate to using ABM for optimising systems rather than for representing human behaviour within operations and service systems. When searching for “social simulation” we found that this was a hot topic in OR but there were only very few papers that mentioned “agent based” in their keyword list. To our surprise we did not find even one paper that mentioned “agent based” in conjunction with UML, a graphical notation that is used to define Discrete Event Simulation (DES) models, in its keyword list.

---

\(^1\) For simplicity, from now on we will only refer to OR, but mean in fact OR/MS

\(^2\) For a brief description of the different simulation paradigms see Siebers and Aickelin (2008)
Table 1 IAOR search results

<table>
<thead>
<tr>
<th>Term 1</th>
<th>Term 2</th>
<th>2006-2009</th>
<th>2010-2013</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulation</td>
<td>System dynamics</td>
<td>73</td>
<td>128</td>
</tr>
<tr>
<td>Simulation</td>
<td>Discrete event</td>
<td>119</td>
<td>93</td>
</tr>
<tr>
<td>Simulation</td>
<td>Agent based</td>
<td>47</td>
<td>85</td>
</tr>
<tr>
<td>Simulation</td>
<td>UML</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>Agent based</td>
<td>UML</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Social simulation</td>
<td></td>
<td>38</td>
<td>83</td>
</tr>
<tr>
<td>Social simulation</td>
<td>Agent based</td>
<td>3</td>
<td>12</td>
</tr>
</tbody>
</table>

In summary, there seem to be some recognition of ABM/S among the OR community but its usage for modelling human behaviour within operations and service systems has not reached the desired extent as suggested in Siebers et al (2010). So what is stopping the OR community from using ABM/S for their purposes? Perhaps it is the lack of “knowing how to” use it. As ABM/S has been widely adopted in in related fields like Business Studies, Economics, or the Social Sciences (Macal and North 2010), the obvious solution would be to adopt well-established approaches from one of the other disciplines. But there are some stumbling blocks that prevent us from doing so. If we consider the different goals we are trying to achieve as well as the different foundations on which we base our models, we can quickly see that it is not a straightforward process to simply adopt an established approach from Business Studies, Economics, or the Social Sciences. Table 2 provides an overview of the differences between OR and Business Studies, Economics, and the Social Sciences in the way they approach simulation modelling and in the way they use simulation models.

Table 2 Different approaches to simulation modelling(adapted from Robinson 2010)

<table>
<thead>
<tr>
<th>Operational Research</th>
<th>Business, Economics and Social Science</th>
</tr>
</thead>
<tbody>
<tr>
<td>Focus on solving real world problems</td>
<td>Focus on understanding the real world</td>
</tr>
<tr>
<td>Data-driven (empirical work)</td>
<td>Theory-driven</td>
</tr>
<tr>
<td>Good quality of data and strong analysis</td>
<td>Sound dynamics hypotheses</td>
</tr>
<tr>
<td>Validation to ensure sufficient accuracy for purpose</td>
<td>Plausibility rather than operational validity³</td>
</tr>
<tr>
<td>Implementation of findings</td>
<td>Learning and understanding</td>
</tr>
</tbody>
</table>

How can we make OR ABM/S more accessible and attractive? One problem that we think limits the success of ABM/S in OR is that the other disciplines that use ABMs for modelling social systems use an equation based approach to create their models which is not compatible with the way we generally do simulation modelling in OR. Ever since the proliferation of visual modelling, we have been using graphical notations in many simulation projects (stock and flow diagrams in System Dynamics Simulation (SDS) and process flow diagram in DES). The extensive use of graphical notations can be seen from many simulation modelling courses offered by OR departments. Therefore our hypothesis is that if a graphical notation can be established, the number of users of OR ABM/S will grow more rapidly. A graphical notation that could be used for ABM/S includes Unified Modeling Language (UML) (OMG 2010b) and Business Process Model and Notation (BPMN) (OMG 2010a). In this paper, we focus on UML since BPMN has been explained in Onggo (2012, 2013). Our current work uses the standard UML (not adding any proprietary extension).

UML has been used in the field of Computer Science but is not well established in OR. The use of a graphical notation makes it easier for the stakeholders who are not an expert in Mathematics to understand and validate the models that are created. This in turn will improve the credibility of the models. So it is important to convince OR practitioners about the usefulness of UML as a modelling approach.

³ For an explanation on "operational validity" see Sargent (2005)
tool. There is also a need to provide some UML training for members of the OR community in order to spread the use of OR ABM/S.

Another problem that we think limits the success of ABM/S in OR is that it is often assumed that agent models require enormous computational power. While this is true for some models it does not apply to all models. If the right level of abstraction is chosen this problem might become less severe. For example, rather than building a model with 1,000,000 complex agents one could consider building a large scale model with 1,000,000 simple agents to capture population dynamics and several small scale models with 1,000 complex agents to study the impact of specific events on human behaviour. Alternatively, population dynamics could be captured by using SDS while the impact of specific events on human behaviour could be studied by using ABM/S. Therefore our hypothesis is that if such considerations are taken into account, ABM/S becomes a feasible modelling option and the use of ABMs could grow more rapidly.

The objective of this paper is to demonstrate how UML diagrams can be used to represent OR agents. We aim to demonstrate that given the right simulation development tool, using ABM/S is not more difficult than using other simulation techniques. The remainder of this paper is organised as follows: Section 2 introduces the notation of UML state machine diagrams and provides a step-by-step guide to developing UML state machine diagrams for OR. Section 3 provides an overview of potential tools to be used for ABM/S in OR. In Section 4 we demonstrate the application of UML state machine diagrams by presenting two conceptual models we built for real world OR case studies. Section 5 provides conclusions and an outlook into the future of OR ABM/S.

2 THE APPLICATION OF UML TO OR ABM/S

The UML is a family of graphical notations that is used in the field of software engineering for describing and designing object oriented software systems (Fowler 2004). The latest UML standard (v2.4) comprises 26 different diagram types. While some of these types are used for capturing the structural design of a software system others are used for capturing the behavioural design of a software system. Often programmers will only need a proper subset of the existing diagrams in order to express the semantics of a large percentage of analysis and design issues (Booch et al 2007). Perhaps the most useful diagrams for ABM/S in OR are use case diagrams, class diagrams, sequence diagrams, and state machine diagrams. A use case diagram is commonly used for the communication between the key stakeholders. They depict who (or what) interacts with the system. A class diagram describes the types of objects (agents) in the system and the various kinds of static relationships between objects. A class can be seen as a template for the agents that will be created. A sequence diagram describes how groups of objects collaborate with each other (by showing the messages passed between the objects) in a specific scenario (use case). A state machine diagram (also called state chart) describes the lifetime behaviour of a single object. They express behaviour as a progression through a series of states, triggered by events, and the related actions that might occur. These diagrams can be very effective in visually capturing the logic within agents and quickly conveying the underlying dynamics of complex models (Ozik et al 2013). In this paper we focus on state chart development although for the case studies (section 4) we will also present some use case diagrams to support the understanding of the state charts.

2.1 UML State Charts

In ABM/S, a state chart can be used to represent the behaviour of an agent at some discrete points in time. An agent will move from one state to another and during the transit, an agent may execute an action. A state chart diagram has a number of graphical elements. The main elements are shown in Table 3.
Table 3  State chart’s main graphical elements

<table>
<thead>
<tr>
<th>Graphical element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Entry pointer: Indicates the initial state after an object is created</td>
<td></td>
</tr>
<tr>
<td>State: Represents a locus of control with a particular set of reactions to conditions and/or events</td>
<td></td>
</tr>
<tr>
<td>Initial state pointer: Points to the initial state within a composite state</td>
<td></td>
</tr>
<tr>
<td>Final state: Termination point of a state chart</td>
<td></td>
</tr>
<tr>
<td>Transition: Movement between states, triggered by a specific event</td>
<td></td>
</tr>
<tr>
<td>Branch: Transition branching and/or connection point</td>
<td></td>
</tr>
<tr>
<td>Shallow history: The state chart remembers the most recent active sub state (but not the lower level sub-states)</td>
<td></td>
</tr>
<tr>
<td>Deep history: The state chart remembers the most recent active sub state (including the lower level sub states)</td>
<td></td>
</tr>
</tbody>
</table>

A simple example of a state chart diagram can be found in Figure 1. A state chart has exactly one state chart entry pointer which indicates the initial state of the agent. A state models a situation during which some invariant condition holds. Usually time is consumed while an agent is in a specific state. A simple state is a state that does not have substates while a composite state is a state that has substates (nested states). A state chart has as many initial state pointers as it has composite states.

In this example, at the top level, the state chart has two composite states. The state on the left has further two substates. It also has a shallow history state. This means after the state chart control has moved from the left composite state to the right composite state, the next time the control moves back to the left composite state, it will remember the last substate it was in before leaving the composite state and will directly jump to that substate. The composite state on the right has one composite substate and one simple substate. It also has a deep history state. The deep history state allows the composite substate to remember the most recent active states including the lower level substates within it. If we would replace the deep history state with a shallow history state, then the composite substate would not be able to remember the simple state it was in before leaving the composite state. Instead it will use the default state pointed to by the initial state pointer. A transition indicates that if the specified trigger event occurs and the specified guard condition is true, the control within the state chart moves from one state to another and performs the specified action (if one was specified). In a state chart it is also possible to define branches, i.e. decision points that determine which transitions to follow next. A final state is the termination point of a state chart. When the control enters a final state, its action is executed, and the state chart terminates. In technical terms this means that the agent object is deleted from computer memory. A more in depth introduction to the UML state chart elements can be found in Fowler (2004) or Booch et al (2007).
2.2 Common OR ABM Model Patterns

Modellers may ask if there are some standard design patterns for ABMs. As such there are no standard design patterns. But there are typical designs that might help you to get going in the right direction and there are designs that often get you going in the wrong direction. Based on our experience, we have seen three common design patterns: centralised design (Figure 2, left side), decentralised design (Figure 2, right side), and hierarchical design (not depicted). The latter is often used in combination with the former design patterns. In a centralised design there is one centre state that one will return to before getting into any other state or loop of states. This state can be thought of as a "contemplating" state that normally does not consume any time. It is a dummy state that is supposed to help simplifying the design of the state chart. But the centre state can also be one that consumes time (e.g. a specific location from which all activities start). An alternative design is the decentralised design where the central state is missing. State charts can also be hierarchical, i.e. each state can contain another state chart, making that state a composite state.

2.3 A Step-By-Step Example of Building an OR ABM

There are many different strategies of getting started building ABMs and it is impossible (and perhaps not very useful) to list them all here. Below, we describe a strategy that often has helped us to get started with designing our OR ABMs. As an example case, we have chosen to model office workers. The development starts with thinking about what actors we want to consider in our models. Once we know the actors, in our example "office workers", we need to define the states they can be in. In
order to come up with potential states it helps to think in terms of "locations" first. For an office worker relevant locations might be: "at home", "at the office", or "elsewhere". The next step would be to think about key "time consuming activities" within these locations. It is important to consider only key locations and key activities as otherwise the state chart gets too complex. One should only define as much detail as is really necessary for investigating the question studied. In our example details of what the worker is doing at home are not really relevant to the study (which focuses on work habits) and therefore the "atHome" state is a single state only, measuring the time the worker spends at home. But the details of the "atOffice" state are relevant to the study (to measure the time the worker is actually working). Therefore, the "atWork" state is modelled as a composite state that contains a sub statechart differentiating the time spent for "working" and "dozing". The result of this modelling process could look like the left side of Figure 3. It shows a centralised design of a worker agent with "atHome" being the centre state. If the modeller wants to consider that the worker should leave the job and go elsewhere directly to work transitions between "elseWhere" and "atOffice" could be added which would make this a decentralised design (as shown on the right side of Figure 3). Once the graphical design of the state chart is finished the modeller has to consider how state changes will be triggered. These could be triggered by conditions, timeouts, rates, or schedules. Finally the modeller needs to consider how the agents will interact, i.e. what kind of messages should be sent and who should receive them and in which situation. Once this conceptual modelling process is finished and the design has been validated by the stakeholders, the implementation can begin. It is important to remember that the whole modelling process described above is an iterative process and that it might take a while to come up with the final design of an agent’s state chart.

**Figure 3 Basic state chart of office workers (left: centralised design; right: decentralised design)**

### 3 UML ABM/S TOOLS

Nowadays there are several simulation modelling tools that support the use of UML state charts as a graphical notation for the modelling process and these tool can automatically translate the graphs into executable code.

Perhaps the most advanced tool is the commercial software AnyLogic (http://www.anylogic.com/) which is a multi-paradigm eclipse based simulation IDE that supports graphical model design for all major simulation paradigms (SDS, DES, and ABM/S). When it comes to the implementation stage, a modeller who plans to use AnyLogic needs to have some knowledge of Java. When a model gets more complex it often requires some programming (e.g. to define a more complex behaviour triggered by an event or to define a more complex transition rules). A very useful feature of AnyLogic is the support of hierarchical model design (e.g. a state chart inside another state chart) and hybrid model design (e.g. SDS and DES sub models in a single multi-paradigm model). Hybrid models can also be hierarchical. An alternative is Repast Simphony...
Siebers and Onggo

(http://repast.sourceforge.net/). It is a free and open source ABM/S platform. In its latest version, it has been equipped with a new agent state chart framework (Ozik et al 2013). Repast Simphony is a java-based modelling system that is designed for use on workstations. A lean and expert-focused C++ based modelling system that is designed for use on large computing clusters and supercomputers is also available but does not support graphical agent modelling (however, it supports a Logo-like programming language which is designed to be simpler than C++). The most commonly used ABM/S tool in academia is NetLogo (http://ccl.northwestern.edu/netlogo/). While it does not support UML (and is in fact not object oriented) it has the advantage that it is easy to learn, as it uses a simple programming language based on Logo. Furthermore many sample models and tutorials exist for this tool, much more than for the other tools mentioned above. There are many other tools but only very few of these support UML based modelling. An up-to-date list of existing ABM/S tools is provided at Wikipedia (2013b).

4 CASE STUDIES

In this section we present two conceptual models that we have developed for real world case studies. Here we focus on the graphical representation rather than on the case studies themselves. Detailed information about the associated case study investigations can be found in Siebers and Aickelin (2011) and Zhang et al (2010). In order to support the UML state chart description we also provide some UML use case diagrams which are commonly used for the communication between the key stakeholders. They depict who (or what) interacts with the system. In these diagram actors are entities that interface with the system while use cases represent what the actors want the system to do for them. The associations in the use case diagram indicate which actors initiate which use cases.

4.1 Case Study 1: Simulating People Management Practices in Retail

In the first case study, we use a combination between ABM and DES to understand the impact of management practices on company performance in a retail environment. In the retail sector (e.g. department store operations) operational management practices are very well researched while people management practices are often neglected by the researchers. The problem seems to be that the tool usually used for such investigations (DES) does not well support the studies of people management practices. In our studies we proposed to use a combination between ABM and DES to model this service system as we have a human centric system with an underlying queuing structure. We use a queuing system for modelling the operations (queues and service priorities within the department store) while we use agents for modelling staff and customers and their behaviours. These agents replace the passive entities traditionally used within the DES system. This is an improvement compared to using only DES as it allows us to model real world human behaviour and consider things like the evolution of customer preferences over time. Also pure DES would not allow us to consider proactive behaviour while ABM/S supports this concept. To demonstrate the usefulness of such an approach we conducted a case study with a top ten UK retailer to empirically inform the modelling and simulation process looking at Audio & TV (A&TV) and Womenswear (WW) departments across two branches.

We started our case study with knowledge gathering through informal participant observations, staff interviews, and informational sources internal to the case study organisation. Then we continued with the conceptual modeling. We decided to consider two types of agents: customers and staff (while staff agents were further subdivided into different types). All staff agents shared a standard design. Figure 4 shows an overview of the conceptual model with customer and staff agents. The transitions are triggered by using frequency distributions (e.g. triangular distributions for determining state change delays) and the decision making was represented using probability distributions. We also considered different types of customers (with different likelihoods to engage in certain activities and different patient levels) by implementing several archetypes for our customer agents. This gave us some control over the behaviour of our customer agents. Using the state chart approach for the conceptual modelling also helped us to communicate the model to our stakeholders (management staff
Siebers and Onggo

at different levels at the case study department store) and supported very well the implementation of this model. For the implementation we used AnyLogic. The archetyping together with the object oriented modelling approach allows us to collect statistics about different types of customers. Our agents were also fitted with a novel customer satisfaction measure which evaluated customer satisfaction at each state transition. Without modelling the customers as agents implementing such a measure would have not been possible.

**Figure 4** Agent state charts (left: customer; top right: staff) and use case diagram (bottom right)

### 4.2 Case Study 2: Office Building Energy Consumption

In the second case study we use ABM to investigate "Office Building Energy Consumption". Each organisation faces a dilemma in terms of energy consumption. It has to provide energy to satisfactorily meet the energy needs of staff and maintain comfort standards in its office buildings but it also has to minimise its energy consumption through effective energy management policies in order to reduce energy bills. Our goal in this case study was to demonstrate the applicability of ABM/S for testing the effectiveness of different energy management strategies. To keep things simple we focused on the electricity consumption. As a case study system we chose our university building. As there is no notion of queues in this model we used a pure ABM/S approach. The purpose of the model was to provide university estate managers with a decision support tool to help them with decisions like: shall we use automated or manual lighting management.

In order to gather some knowledge we consulted with the school's director of operations and the university estate office. We also conducted a survey amongst the school's 200 PhD students and staff on electricity use behaviour. From the survey results we were able to identify some archetypes regarding working hour habits (early birds; timetable compliers; flexible workers) and energy saving awareness (environment champion; energy saver; regular user; big user). We decided to consider four
different types of agents: energy user (including the PhD students and staff), computers, lights, and offices. While energy users are considered as active entities the other three types of agents are considered as passive entities. Figure 5 shows an overview of the main statechart diagrams. For the design of the energy user agent we followed the design principles of the worker agent mentioned in Section 2.3. First the location states are defined and then time consuming activities are defined within the location states.

**Figure 5** Agent state charts (left: energy user; top right: computer, light) and use case diagram (bottom right)

## 5 CONCLUSION

In this paper we have demonstrated how the use of a graphical notation can simplify the life of an OR practitioner (both, academic and industry) who want to enjoy the benefits of using ABM/S. We strongly believe that ABM/S has the potential to become the predominate paradigm for modelling human behaviour in human-centred systems in OR in the near future. Suitable tools do already exist but what is missing is the promotion of these tools in the OR community (see also Siebers et al 2010). Training should be organised to enable people to take advantage of this technology.

Our next step will be to look at other relevant UML diagrams (use case diagram, class diagram, and sequence diagram) as they will allow us to provide a more complete specification of our agents, their interactions, and the environment in which they live in. This should then help to improve the conceptual modelling process (for example, to communicate the model to people who may not be familiar with computer codes) and the validation of the models as it allows us to better explain the structure and behaviour of our simulation models. The co-author has also worked into the use of Business Process Modelling Notation (BPMN) in representing ABMs. This notation is similar to the UML state chart notation but might be more familiar to people from Business Schools and therefore
Siebers and Onggo
easier to use for OR studies. Unfortunately, there is no ABM/S tool that supports BPMN at the moment. However, a prototype has been developed to show that this is possible (Onggo and Karpat 2011).

ACKNOWLEDGMENT
We would like to thank Gaurang Phadke (Business School, Bayreuth University, Germany) and Holger Schnädelbach (School of Computer Science, Nottingham University, United Kingdom) for their help and we also very much appreciate the valuable comments from the reviewer.

REFERENCES
AUTHOR BIOGRAPHIES

PEER-OLAF SIEBERS is a lecturer in the School of Computer Science at the University of Nottingham in the United Kingdom (http://www.cs.nott.ac.uk/~pos/). His main research interest is the application of data driven computer simulation to study human-centric complex adaptive systems. This is a highly interdisciplinary research field, involving disciplines like social science, psychology, management science, Operational research, economics and engineering. Other areas of interest include Risk Analysis and Systems Biology.

BHAKTI S S ONGGO is a lecturer in the Department of Management Science at the Lancaster University Management School, Lancaster, United Kingdom. His research interests lie in the areas of simulation methodology (modelling paradigms and conceptual modelling), simulation technology (parallel and distributed simulation, web-based simulation cloud-based simulation), business process modelling and simulation applications. His email address is s.onggo@lancaster.ac.uk.