The Role of Cloud Computing

Ash Ings-Lamb and Alex Voss
University of St Andrews
Procter, Rob
University of Manchester

Table of Contents

1 Executive Summary

The use of cloud technologies to teach agent-based modelling and simulation (ABMS) is an interesting application of a nascent technological paradigm that has received very little attention in the literature. This report fills that gap and aims to help instructors, teachers and demonstrators to understand why and how cloud services are appropriate solutions to common problems they face delivering their study programmes, as well as outlining the many cloud options available. The report first introduces social simulation and considers how social simulation is taught. Following this factors affecting the implementation of agent-based models are explored, with attention focused primarily on the modelling and execution platforms currently available, the challenges associated with implementing agent-based models, and the technical architectures that can be used to support the modelling, simulation and teaching process. This sets the context for an extended discussion on cloud computing including service and deployment models, accessing cloud resources, the financial implications of adopting the cloud, and an introduction to the evaluation of cloud services within the context of developing, executing and teaching agent-based models.

The core discussion on cloud computing, which aims to act as an introduction to those aspects of the cloud most directly relevant to teaching social simulation, is followed by an extended comparison of the Amazon Web Services' infrastructure Elastic Compute Cloud (EC2) and the Google Cloud Platform's Google Compute Engine (GCE), another infrastructure cloud offering, as well as brief discussion of other infrastructure clouds. The comparison here is unavoidably one-sided simply because AWS is by far the most dominant cloud market player and information on its services and products is abundant, readily available and easy to access, in the same way that its cloud services are abundant, readily available and easy to access. Over time this situation is likely to change as other cloud service providers mature and erode Amazon's market dominance, but at the time of writing it is likely that ABMSs that make use of public clouds will do so using EC2 in preference to other public infrastructure clouds.

Following the comparison of cloud providers the paper introduces the core use cases identified as of highest utility to those running agent-based simulations in the cloud, consisting of six different ways that the cloud could be used to execute agent-based models and simulations. How these use cases relate to teaching social simulation using cloud resources succeeds this section, where we identify five core arguments that motivate the use of cloud to teach social simulation, before exploring how teachers might go about choosing which cloud resources to use and how to do so. Having given readers a reasonably complete overview of the potentialities associated with adopting the cloud we then describe our experiences of using EC2 to realize several of the use cases previously identified, including an outline of the steps required, problems and difficulties encountered, and the results of our simulation runs. Finally we conclude by considering options for future work, including possible future research directions and projects that could significantly reduce the effort required to leverage the cloud for teaching ABMS, and increase awareness of the options open to both students and instructors. There are also appendices, which give

extra technical information on aspects of the cloud and practical step-by-step walkthroughs of running agent-based simulations in the cloud. The report also contains a number of recommendations that are closely related to our considerations on future work, and which aim to guide students, instructors and other interested parties in their investigation, adoption and use of cloud technologies for studying, teaching and executing agent-based models and simulations.

2 Introduction

This report is an introduction to the use of cloud computing for teaching agent-based modelling and simulation (ABMS). It is designed to act as a decision support document for instructors planning an ABMS study programme and aims to equip them with the information required to make informed decisions about the types of cloud technologies available to support ABMS teaching, along with the challenges and benefits associated with different aspects of the cloud. We hope that the information contained in the report will be of use across a wide variety of teaching contexts including self-paced courses (typically online), distance learning, conference tutorials and workshops, university lectures, seminars and tutorials (including face-to-face and Web-based workshops, mixed-mode courses incorporating distance learning, webinars and webcasts, face-to-face tutorials, etc.) and professional short courses.

Cloud computing provides numerous opportunities for those teaching ABMS. How cloud technologies are adopted turns on a number of factors because 'the cloud' is an amorphous, ill-defined concept spanning numerous technologies designed to deliver computing resources and services in different ways in different contexts to meet different user requirements. Infrastructure clouds, referred to as Infrastructure-as-a-Service or *IaaS* clouds, such as Amazon's Elastic Compute Cloud (EC2), provide on-demand access to low-level computational resources such as CPU core (processing power), storage and networking, across the public Internet. Platform clouds –Platform-as-a-Service or *PaaS* – such as Google App Engine and Microsoft Windows Azure provide users with an online application development environment and associated tools support. Finally Software-as-a-Service (SaaS) allows users to provision software applications across the network. A simple example is the use of Gmail to deliver university email services such as the University of St Andrews' student email service. Other common SaaS platforms implement office, collaboration and end-user productivity applications such as Google Docs and Microsoft Office 365.

Considering scope this report focuses on IaaS clouds because the services they provide are directly applicable to the delivery of ABMS study programmes. A number of powerful though simple arguments motivate the integration of infrastructure cloud technologies with ABMS teaching and delivery strategies. One clear benefit for instructors is the ability to dynamically provision and scale cloud resources according to short-run computational requirements. This gives instructors and demonstrators the flexibility to deploy high-end, powerful, and fully configurable compute resources on-demand and at short notice, detaching the successful delivery of their

programmes from local technology constraints, and giving them the agility associated with a self-service on-demand computing model.

Running an agent-based model in the cloud using free/open source software will require payment for an appropriate machine instance type, data storage, and the costs of transferring data into and out of the cloud.

2.1 Scope

We restrict our focus to simulations that only require the computational resources of a single desktop or laptop computer. We do not consider in detail issues relating to running simulations that require high performance compute resources. Modelling and execution platforms such as Repast for High Performance Computing are briefly considered, and for completeness we touch on ways the cloud could be used in high performance scenarios, but in the main our attention is directed to issues relevant to the type of small-scale models and simulations typically developed and run in a beginners to intermediate course. We do though look at how high throughput simulation jobs can be executed in the cloud. In particular we compare running batch mode simulations using the Jenkins continuous integration build server as a task management front-end controlling worker compute nodes in the cloud, with the Model Exploration Service, a web service designed to allow users to run batch simulations on EC2. This complements our discussion of two more practical demonstrations introduced in section [x]. The first details the relatively straightforward task of manually configuring a virtual machine running on EC2, whilst the second shows how the same simulation environment can be installed and configured using a contextualization script. Contextualization scripts are scripts designed to run after booting a virtual machine in the cloud and replace the need for manual user configuration. One recommendation we make is for the development of a version-controlled repository of simulation-specific contextualization scripts that instructors can access to easily setup different types of simulation environment in the cloud. This avoids the need to maintain a repository of preconfigured virtual machine images, instead giving instructors the option of modifying existing scripts to meet their requirements if a suitable script is not in the repository (or writing a new script if necessary).

After giving a high-level outline of the services constituting the cloud ecosystem we also confine our attention to Infrastructure-as-a-Service (IaaS) clouds, as they are most likely to be of use and interest to present and future academic research into social simulation, agent-based modelling and teaching. De Oliveira, Baiao, and Mattoso (2010) expound the main advantages associated with cloud computing as following from the relative ease with which users can access a wide variety of computational resources with a minimum of configuration, and without high capital acquisition costs. Similarly the ease with which instructors can access infrastructure cloud services such as EC2 before, during and after lessons gives them great flexibility and opportunities to experiment with different computing technologies.[1]

---

[1] De Oliveira *et al*. also discuss a number of desiderata for using the cloud to run scientific experiments, covering areas such as reproducibility, validation, the

Users of IaaS clouds can typically formulate solutions to their problems at a machine level by specifying numbers of processor cores, available memory, CPU architectures and speeds, and storage requirements. Therefore Iaas cloud services allow instructors flexible and convenient access to compute and storage resources during the preparation and delivery of ABMS study programmes such that is not possible when relying on in-house, local resources or distributed, remote grids and clusters. In part this is a function of often-limited local hardware and the complexity of running jobs on the grid, and in part a function of the unique opportunities proffered by the development of high-end, dynamically and rapidly scalable, self-service access to massive computational power and storage capacity that has been enabled by developments in the cloud. Furthermore, the costs of accessing public clouds such as EC2 are falling. This means that instructors (and students) have the ability to run computationally expensive jobs and store the generated artefacts with only minimal financial outlay, a good Internet connection, and local commodity hardware (desktop, laptop, mobile, etc.) in a way not previously possible. The development of private clouds by universities and other institutions, as well as increased use of hybrid and community clouds, offers similar opportunities.

The ease with which IaaS cloud services can be rapidly scaled, vertically and horizontally, to meet expensive computational demands also makes the cloud useful for running simulations that require high performance compute resources that use dedicated software packages such as Repast for High Performance Computing (introduced below). Using the cloud for high performance simulation jobs is though outwith the scope of this report as we focus on those aspects of the cloud that are most useful to teachers in typical ABMS teaching contexts focused on introducing the principles and techniques of agent-based modelling to beginners through small scale simulations executable on commodity compute resources.

One key advantage of the cloud is the ability to customize the type of services provisioned, and to rapidly change these services in line with changing requirements. This is potentially very useful in the context of running simulations and teaching social simulation. To effectively leverage this potential, instructors should understand the variety of cloud services available, their interrelations and how these resources are sourced and deployed. Hence we also look at different cloud deployment models. Deployment models categorise how cloud services are sourced, and consist of public, private and hybrid clouds. There are advantages and disadvantages associated with each deployment type, depending on the specific use case, and we look at those relevant to teaching ABMS.

We also introduce the technologies that enable the cloud so that instructors have some idea of how the cloud works and the constraints these technologies can impose. Virtualization is the key technology enabling multi-tenant architectures,

---

provenance of final and intermediate results, and the parameter values used to generate results, all of which is useful reading for teachers and students of ABMS.

giving cloud users the ability to rapidly provision on-demand servers, and provide the illusion of infinite available compute capacity. Web service calls are used to access cloud resources across the network, which can be programmatic, through the command line, or via GUI management tools. Clouds also rely on middleware platforms to coordinate resource and access management and we briefly introduce these too. Our goal in discussing these technologies is to help potential cloud users understand how the cloud works without going into great detail, which is unlikely to be of great interest to readers of this report.

There is very little literature specifically addressing teaching agent-based modelling and simulation. Friesen, Laskowski, Demianyk and McLeod (2010) for example discuss the educational opportunities associated with agent-based modelling and simulation, with particular emphasis on engineering students. They believe that the development of agent-based modelling as a paradigm within the broader field of social simulation has the possibility to provide unique educational opportunities for undergraduate and postgraduate students, but this is only a brief conference paper and in general this type of work is rare in the literature. Macal and North (2010) is an exception and below we outline their work in this area as a point of comparison with the experiences of other teachers that we discuss later in the report. They refer to the "agent-based modelling conundrum", whereby high numbers of people are drawn into agent-based modelling, attending courses designed to acquaint them with the core knowledge and skills necessary to pursue their own modelling interests, without any of the prior knowledge or experience necessary for success. This gives instructors many difficulties as they attempt to widen as far as possible access to courses on agent-based modelling and simulation, while accommodating classes composed of students with varying skill sets, technical aptitudes and background experiences. The approach taken to teaching ABMS is dependent on these factors, meaning instructors should vary course content relative to the skills and aptitudes of their students. Here the cloud offers great flexibility in the supply of computational resources not open to those relying on local commodity hardware, whether supplied by local institutions or brought to class by students. Flexibility in resource provisioning allows for the detailed production of course and lesson plans that are specifically tailored to their audiences, taking account of skill levels, the course syllabus and teaching objectives.

3 Recommendations

The report makes a number of headline recommendations to instructors who are considering the use of cloud technologies with their courses on ABMS. These are:

I. **Recommendation:** Version-controlled ABMS contextualisation script repository
II. **Recommendation:** E-Learning System
III. **Recommendation:** Another
IV. **Recommendation:** Number 4
V. **Recommendation:** Five should be enough…

[Come back to this]

Considering deployment models, instructors have a basic choice between either public or private clouds. The quality of service users receive from private clouds will vary with the institution running them but the major public clouds such as AWS, Microsoft Azure or Google Compute Cloud are easy to access, dependable, and relatively cheap, and can easily meet the computational demands of the most complex and large scale simulation runs, but clearly the final decision will vary from case to case. The configuration effort needed to properly use either public or private clouds will always be non-trivial but, on a case-by-case basis, should not present great difficulties.

[Expand here]

4 Social Simulation

In this section we introduce social simulation and the logic that underpins the modelling of social systems. We then introduce a particular approach to social simulation – agent-based modelling – that is currently receiving intense research interest. For expository purposes the remainder of the paper focuses directly on the use of cloud technologies to develop, execute and teach agent-based modelling, although readers should easily see that our remarks are applicable, to teaching other types of social simulation.

4.1 Methodology

Social simulation studies socio-economic phenomena by investigating the social macrostructures and observable regularities generated by the behaviour and relationships between individual social agents, and the environment in which they act. Typical macro-structures studied via simulation include tangibles such as organisations – governments, companies, educational institutions – and intangibles such as markets, as well as more abstract phenomena – the prevalence of social norms, population distributions, epidemic dynamics, and the emergence and establishment of socio-economic class systems within society.

From a methodological perspective social simulation occupies a space between theoretical social science and experimental empirical investigation. It is an interdisciplinary field sitting at the intersection of social science, agent-based computing, and computer simulation (Davidsson 2002).[2] By developing models of complex social systems and using computers to study their evolution through simulated time, researchers have a laboratory where they can observe the interactions between social agents and processes, and have the opportunity to test hypotheses in a way not possible before the advent of modern modelling and simulation techniques (Li *et al*. 2008).

[2] It is possible to conduct social simulations without computers but we do not pursue such techniques in this report.

This approach yields opportunities to investigate complex, nonlinear dynamical systems that are resistant to analytic statistical explanation based on linear relationships that hold between the dependent and independent variables of a system (Gilbert and Troitzsch (2005)). Social simulation mitigates the trade-off between mathematical tractability and the requirement to incorporate simplifying assumptions into models of social processes that are necessary when developing the static models traditionally used for prediction and explanation. Autopoietic systems, studied in self-organization theory, and characterized by the ability of the units comprising the system to self-produce and self-maintain, also become viable targets for exploration using simulation tools and techniques. The concept of emergence also plays an important role in the systems studied using simulation techniques, feeding causal explanations of how complex behaviour, structures and social processes result – emerge – from less complex (relatively simpler) actions, behaviours and local interactions of individual, heterogeneous agents.[3]

4.2 Types of Social Simulation

There are many types of social simulation (see Gilbert and Troitzsch 2005 for a convenient introduction). System dynamics is one approach that encodes the properties and behaviour of a target system as a set of logico-mathematical equations from which future states of the system are analytically derived. Agents are considered indistinguishable and homogenised, and their inter-relationships static, unable to change over simulated time. Static and dynamic micro-analytical simulation techniques in part answer this limitation. Static micro-simulation introduces agent heterogeneity by populating models with distinct, individual agents, and splits the target system into different levels in the model system. Dynamic micro-simulation permits changes to target system populations to be represented in the system model and thus allows the model to exhibit dynamic behaviour over simulated time. Other types of simulation include queuing models, cellular automata, and evolutionary simulation models, as well agent-based models, which we concentrate on for the remainder of the paper.[4]

4.3 Agent-based Modelling

Agent-based modelling and simulation (ABMS) forms the core of Epstein's "generative social science". The goal of generative social science is to "grow" artificial social structures in order to study the links and relationships between local, heterogeneous, individual agents and the macro social structures they generate. Individual people in the domain or society under study are represented as agents that interact locally through processes that are social relevant, including birth and death rates, violence, disease, crime, and warfare, as well as the externalities produced by other social actors such as industrial pollution (a negative externality) or positive health externalities (Li *et al*. 2008).

---

[3] See Epstein (2006) for an interesting discussion of emergence in social simulation.

[4] Gilbert and Troitzsch (2005) is a convenient introduction to the many types of social simulation.

One hallmark of ABMS, responsible in part for the high computational cost necessary to simulate the models, is the direct map between agents in the model and social actors in the real-world domain, with each constituent represented one-to-one by an agent in the model. Each agent has its own attributes and behavioural properties, and interacts with other actors and the environment according to a collection of rules that codify the state transitions of agents and the environment over time, which is controlled by an event scheduler. ABMS schedulers typically implement either time-step or discrete event scheduling, with the latter approach more appropriate as models grow in size and complexity (North and Macal 2007). To model the interactions between agents, ABMS makes use of topologies to define the notion of a 'local space' or 'neighbourhood' (Macal and North 2009). This neighbourhood delimits a local area within which agents can interact and thus controls the extent to which agents exchange information through limiting inter-agent activity to between only those agents within a given neighbourhood at a given time. In this way the dynamics responsible for the social phenomenon under study are partly generated through specification of the topological structure of the model.

In constructing a simplified model of a target social phenomenon, individual agents, their attributes, and environment are specified. Like the target system, the model is dynamic, having both structure and behaviour, because it aims to capture real world dynamic social systems that evolve over time. To realise dynamic behaviour agents in the model are specified in detail with attributes and first-order behavioural rules governing their interactions with other agents, the ability to conduct sophisticated decision making over simulated time, and are given second-order behavioural rules governing changes to first-order behavioural rules, memory and available resources, (Macal and North 2009). The model structure is specified by a set of parameters specifying the initial conditions of the model. The detail required in the specification of agents and their environment is necessary to generate complex, dynamic and adaptive behaviour precludes the use of statistical techniques used to construct static models of linear systems. Linear systems analysis permits analytic techniques that derive the future structure of the system model because linear systems are represented as sets of equations admitting tractable mathematical resolution using statistical and algebraic methods. The nonlinear social systems studied using simulation techniques are too complex to accommodate analytic approaches. Instead, computer simulations representing the evolution of the model's structure are used to study how the target system reacts under permutations to its parameters and initial conditions.

4.4 The Workflow and Logic of Social Simulation

There is no standardized way of approaching social simulation or the construction of complex agent models. Gilbert and Troitzsch (2005) explicate the development process as broken into distinct stages including model design, model construction, verification and validation, and publication. Grimm and Railsback (2012) give a number of heuristics to aid the modelling process such rephrasing the problem statement, using simple diagrams to understand the model's structure, imagining you are inside the system, and identifying essential variables and simplifying

assumptions. They go on to formulate a six stage modelling lifecycle consisting of formulating the question under study, assembling hypotheses to test, determining the model structure, implementing the model, performing analysis, and finally communicating the results. In general, after identifying an explanatory target – a given social process, outcome, macro structure or configuration – an abstract representation is constructed to model the target. The model is then run through simulated time to produce simulated data, paralleling the act of gathering empirical data on the target. This forms the basis of an analysis and a process of identifying connections between the simulated and collected data in order to verify that the model produces results similar to those of the real world.

5 Teaching Agent-Based Modelling and Simulation

There is very little literature specifically devoted to teaching agent-based modeling and simulation. Macal and North (2010) though do discuss the elements of a typical course on ABMS. We outline this syllabus below as a point of reference and comparison with the experiences and problems encountered by ABMS teachers and demonstrators.

5.1 Macal and North's ABMS Teaching Syllabus

Macal and North give three high level objectives when teaching agent-based modelling and simulation. These cover how to think about ABMS, how to do ABMS, and the aim of furnishing students with a language for engaging with and discussing ABMS. To meet these objectives they describe a collection of teaching techniques and strategies, and discuss the structure of an ABMS teaching syllabus that can be calibrated in line with the skills of the audience and the length of the course. They stress the diversity often found within cohorts enrolled on ABMS programmes (the ABMS conundrum), the variety of skills and knowledge required to successfully pursue serious work in ABMS, and links between this diversity and the challenges inherent in teaching ABMS.

Macal and North's paper is split into four sections. The introduction draws attention to the wide range of applications to which agent-based modelling is applied, where they mention ecology, biology, market analysis, supply chains, military planning and economics and more. They also explicitly distinguish between what they term agent-based simulation (ABS), system dynamics (SD) and Discrete Event Simulation (DES), and they stress that although these approaches to simulation share many similarities there are also significant differences which makes the incorporation of agent-based simulation into other, non-agent-based courses problematic. Factors that make this difficult include the varying computational platforms used to model systems and execute simulations, the backgrounds and skill sets of students studying different types of simulation– including differences in programming, quantitative, modelling, natural and learned skills and aptitudes – and the different problem domains in which different types of simulation are used.

Section 2 reflects on developing ABS teaching strategies. They begin by highlighting the wide degree of interest in agent-based approaches, from both students and instructors, which they contend derives from the fact that we are all ourselves

agents, and because agent-based modelling deals with identifiable agents in place of idealized abstractions or representative values, it is possible to generate a level of student engagement and enthusiasm for agent-based modelling that does not exist with other simulation approaches.

They next allude to the history of ABS, citing in particular work in complex systems, complex adaptive systems, artificial intelligence and the evolution of cooperation. It is these deep, wide ranging intellectual antecedents, combined with multifarious problem domains that results in such a broad range of students studying agent-based modelling and simulation. Variation in student backgrounds, from areas such as computer science, sociology, psychology, philosophy, environmental and political sciences, and engineering, mean that instructors must strike a balance between ensuring that a course on ABMS includes at least a rudimentary introduction to each of the key topics central to agent-based simulation, and at the same time hold the engagement of students with more advanced skills and deeper background knowledge. Students, in the experience of Macal and North, tend to have some but not all of the skills necessary to pursue agent-based modelling, although these skills are typically developed in other areas.

Macal and North distinguish next between introductory courses on ABMS, which focus on *modelling complex systems*, and more advanced courses focused on *building complex system models*. Introductory courses on modelling complex systems are focused on teaching users the basic principles of modelling in general, as well as agent-based modelling in particular. Advanced courses, which focus on the use of a single modelling and execution platform, are further broken down into courses aiming at *basic proficiency* with a particular platform and tool set, and *large-scale agent-based simulations*, which, as its name suggests, focuses on developing larger models that utilize the full range of functionality provided by a given software package. They note however that it is possible to include practical modelling work in introductory courses, and they expect students having attended introductory courses containing practical modelling work to have the skills necessary to create their own models independently. At the other end of the spectrum courses focusing on large-scale ABMS are designed, on this view, to develop the skills and capabilities of more advanced students by, for example, constructing models that scale agent numbers, deepen agent behavioural complexity and memory, and extend the complexity and detail of the simulation environment in which model agents operate.

Macal and North list a number of tools available to instructors to achieve their objectives beyond the software packages we discuss below, designed specifically to facilitate agent-based modelling. These include:

- Distributed computer technology
- Artificial intelligence and machine earning (neural networks e.g.)
- Geographical information systems (GIS)
- Database systems
- Version control systems (VCS)
- Integrated development environments (IDE)

In section 3 Macal and North give a general ABMS course outline, discuss practical demonstration work, student skills, measuring student progress and achievement, and give two course syllabi, intended for week and semester long courses.

Their outline of a general course on ABMS includes a set of core topics designed to give students a comprehensive introduction to modelling, which, they say, goes into far more detail than would a course restricted to learning to use a specific tool or software platform. They include a twelve-point list of topic areas, noting that topics 1-7 relate specifically to agent-based modelling and simulation whilst topics 8-12 are applicable to a more generalist course on modelling and simulation. We reproduce the list here for convenience and reference:

1. Introduction and basic ABMS concepts
2. ABMS use cases and why ABMS is appropriate for a given problem domain
3. ABMS model design process
4. Comparisons of differing ABMS modelling methodologies
5. Tools for ABMS
6. Detailed examination of one (two) ABMS tools in particular
7. ABMS model architectures
8. Verification and validation procedures
9. Data collection and cleaning
10. Output analysis
11. Results presentation
12. Project management

Macal and North also discuss the power and relevance of including practical in-class demonstrations that motivate students, provide introductions to core concepts and techniques, and give students ideas on the types of models and simulations they might like to construct themselves. The time spent before and after practical demonstrations is also important. Prior to a demonstration students must be introduced to the system under study and be familiar with the rules governing agent behaviour and the simulated environment. This allows students to make predictions about expected results and system behaviour before a simulation is run. Expectations can then be compared with actual results and the behaviour of the system can be analysed with respect to the modelling and systems concepts taught during the study programme. The six example simulations, cited because they expose students to a wide range of modelling applications, which are suitable for in-class demonstration, are, amongst many others:

1. Conway's Way of Life
2. Boid's Flocking Simulation
3. Schelling Housing Segregation Model
4. Mass Opinion Spreading Simulation
5. Matching Triangles Participatory Simulation
6. Beer Game Simulation

As regards skills, Macal and North draw upon work by the Project Management Institute, listing seven separate skill sets, in varying degree taught and presupposed by instructors. They make the basic though important points that the level to which a particular skill set is taught is dependent on (I) the skills background of the course participants, and (II) the course time available. Furthermore, course prerequisites are determined by the skills on the list that are not taught during a particular study programme – if a skill is on the list, and isn't taught during class, it should be consider a prerequisite. We again reproduce Macal and North's key skills list for convenience:

1. Modelling
2. Programming
3. Model verification and validation
4. Data collection and cleaning
5. Model analysis
6. Results communication
7. Project management

Considering the measurement of student progress and achievement Macal and North give an eleven-point list that maps almost directly onto the twelve-point list of core topic areas discussed above. Unsurprisingly this includes the ability to describe the modelling process, compare different approaches to modelling, describe different use cases, have knowledge of the model design and development process, develop modelling-specific project management skills, demonstrate the use of one or two model development and execution software platforms, perform basic model verification and validation, understand the techniques and problems of data collection and cleaning, discuss model output analysis, and at least describe, if not display, the ability to present output results to an audience of decision makers.

The final part of Macal and North's paper outlines two teaching programmes, designed for courses lasting one week and one semester. The former is quite detailed, containing a 9am-5pm breakdown of topics covered on each of the five days, and gives start and end times for the different topics. They also suggest that this course could be broken down into courses lasting only three or just one day. The latter, longer course is correspondingly more detailed, covering the range of objectives detailed above.

5.2 Experiences and Problems Teaching ABMS

In this section we compare Macal and North's work on teaching ABMS with first-hand reports of experience teaching ABMS within the contemporary UK higher education system, although this is not confined to formal undergraduate and postgraduate teaching.

[Reports/experiences/conversations/other info we can gather to go here]

6 Implementing Agent-Based Models

In this section we discuss the implementation of agent-based models. We focus on the software platforms and technical architectures available to support ABMS. Agent-based models sit on a continuum from small-scale exploratory models with idealized assumptions that focus on the most important or specifically interesting elements of the target system, to large-scale decision support applications incorporating substantial empirical data aimed at supporting policy formation and high-level decision making (Macal and North 2009). This determines the most suitable software to use. A wide range of development and execution platforms is available, including general-purpose software packages supported by mainstream programming languages, and special purpose toolkits specifically designed for implementing agent-based models. Scale also determines appropriate hardware, with small-scale simulations requiring only the resources of a typical desktop computer, while very large simulations necessitate the use of high performance compute clusters, grid technologies or cloud resources.

Literature on modeling platforms is plentiful. For example Nikolai and Madey (2009) provide an in-depth comparison of ABMS modelling and execution packages, while North and Macal (2012) contains extended discussion of the technical aspects involved with agent-based modelling. Grimm and Railsback (2012) also contains a profitable discussion on modelling environments, and although their primary focus is on ecology the first three chapters on agent-based modeling are an excellent introduction to the field. Macal and North (2007) provide a useful, loose taxonomy of the types of tools necessary to run simulations at different scales and complexities. Of most interest to ABMS teachers are platforms used to run relatively simple simulations at the less complex end of the modelling spectrum. Basic commodity hardware – desktop or laptop computers – is generally sufficient at this scale. This type of modeling platform includes spreadsheet applications such as Microsoft Excel (with VBA programming), prototyping environments such as Repast Simphony, NetLogo and StarLogo, and mathematical modeling platforms such as MATLAB and *Mathematica*. Each is an appropriate candidate for teaching ABMS, although the integrated development and simulation tools incorporated into Repast-like prototyping environments make them by far the most suitable option for introductory to intermediate courses.

6.1 Development and Execution Platforms

An ABMS development and execution platform is a software package allowing users to develop (code) and run (execute) agent-based models. The choice of platform used for a typical course on ABMS is likely to be relatively straightforward because in most cases a graphical development interface supported by an appropriate runtime environment is sufficient.

Although each platform has unique features, and is designed with more or less specific uses in mind, readily available, open source platforms such as Repast, MASON, Swarm and NetLogo are all suitable options. A range of factors affect the choice of platform including the size and complexity of the target system and model, the real world domain under study, goals of the course – introductory overviews,

detailed case studies, technical skills development – target audience, including undergraduates, postgraduates, social scientists, software developers, academic staff, researchers, and professionals, length of course, scheduled contact time, available computational resources such as desktops, laptops, grids, clusters and clouds, skills available and required of both instructors and students, and the academic, educational and professional backgrounds and interests of the instructors and students involved.

6.1.1 Spreadsheets

Spreadsheets are the easiest though most restrictive software tool available. Developing simple models of agent behaviour is relatively straightforward using spreadsheets, considered against the complexity of dedicated packages, but suffer from concomitant problems relating to complexity – limited diversity and interactivity of and between agents – and the scalability, of models they permit. Despite this Macal and North (2007) cite Bower and Bunn (2000) as developing useful agent-based models using spreadsheets. Example packages include Microsoft Excel with VBA programming and other spreadsheet applications such as OpenOffice, which also has (limited) support for VBA, and LibreOffice, which allows users to implement macros using the Basic programming language. We include spreadsheets for completeness though it is unlikely that readers of this report will intend to use them to teach ABMS.

6.1.2 Computational Mathematics Systems

Macal and North (2007) highlight the usefulness of computational mathematics systems (CMS) in developing agent-based models, especially if users have a background using such tools (*cf*. Thorngate 2000). They state that the CMS must implement a fully-fledged scripting language and support array (list) processing. The former is necessary to sidestep the compilation and linking entailed by compiled languages such as C++, while array processing is necessary for efficiently executing simulations. The benefits conferred by a CMS stem from the interactive user interface, allowing the modeller to closely explore the model during development, and the extensive range of mathematical functionality available. Examples include MATLAB, Mathematica, Gauss and Forio Simulate. As with spreadsheets it is less likely that a CMS will be used to teach ABMS than a dedicated modelling environment such as Repast Simphony.

Mismatch between these platforms and agent models. Natural expression of agent based models using object oriented langauges.

6.1.3 Object-Oriented and Procedural Programming Languages

Macal and North highlight the usefulness programming languages such as Java, C++, Objective C and Python in constructing simulation models. It is possible to use procedural languages such C to implement agent-based models but languages such as Java, supporting object-orientation, are far more helpful because they allow

agents in the domain to be modelled by objects in the language.[5] Other features particular to object-oriented programming useful for implementing agent-based models include inheritance, polymorphism and encapsulation. Students learning ABMS will most likely have to gain at least passingly familiarity with one or more major object-oriented language, or dedicated modelling language based on a mainstream language, although the proliferation of dedicated modelling environments means this will probably be one component of a larger development package.

6.1.4 Dedicated Prototyping Environments

Dedicated prototyping environments such as Repast Simphony are available to run small-scale simulations. Geared towards enabling beginners or those with only limited experience and skills to learn the basic principles of ABMS over short time frames, they are also useful for users with greater technical proficiency (Macal and North 2007). The models they permit are more complex than those developed using spreadsheets or CMSs, but can be easily and quickly designed, implemented and run, making them suitable to use over a three-day or other short course on ABMS. Examples include the Repast suite, StarLogo and NetLogo. Instructors interested in using the cloud will most likely do so using a prototyping environment such as these because they bundle all of the tools students need to implement their own models. We use Repast Simphony running in the cloud in the technical demonstrations described in section [x].

6.1.4.1 Repast Simphony

Repast is an acronym for the REcursive Porous Agent Simulation Toolkit, a collection of modelling and execution platforms featuring support for Java, Python and the .NET framework – Repast J, Repast Py, Repast.NET. Repast Simphony[6] is a fourth member of the Repast suite, a cross-platform Java-based modelling package designed for use on small-scale local resources including desktops and local clusters. It has received extensive treatment in the literature. One advantage of this package is that it does not dictate how models are implemented. Models can be directly coded in Java or Groovy[7] (a modern scripting language, fully integratable and interoperable with existing Java code, designed to run in the Java virtual machine. It adds the benefits of modern dynamic languages (e.g. dynamic typing) while retaining access to the Java class library). ReLogo is another option available with Repast Simphony. ReLogo is a dedicated modelling environment and language integrated with the Eclipse IDE that is bundled with the Repast Simphony download. Repast Simphony also has point-and-click tools to generate detailed flow charts mapping processes and agents in the domain onto agents and processes in the model.

---

[5] Python supports both procedural and object-oriented programming paradigms, and is an excellent language to learn the principles of modelling and computer programming.

[6] http://repast.sourceforge.net/repast_simphony.html

[7] http://groovy.codehaus.org/

6.1.5 Large-Scale Dedicated Modelling Environments

In general agent-based models built using spreadsheets, small-scale prototyping environments and CMSs can be developed and run using local desktop machines typically available through university departments. As simulations scale and gain in complexity, other modelling environments are apposite, including Repast for High Performance Computing, MASON, Swarm and AnyLogic.

6.1.5.1 Repast for High Performance Computing

[Paragraph on RHPC here?]

6.1.5.2 Swarm

Swarm[8] is a platform-neutral collection of software libraries for the development of modelling and simulation software, consisting of a development framework, libraries, and a community user base (Grimm and Railsback (2012, chapter 8) is an easy to read and helpful introduction to issues surrounding the development of software for agent-based models and contains a good discussion of the Swarm framework).

[Expand?]

6.1.5.3 MASON

MASON[9] is a Java-based simulation platform providing the core components necessary to build and run large-scale multi-agent simulations, and includes a modelling library and visualization tools.

[Expand?]

6.1.5.4 AnyLogic

[Paragraph on AnyLogic here?]

6.2 Implementation Challenges

[Not sure if this section is relevant or not]

Many general and technical issues arise implementing computer-based agent-models in a classroom context. Broadly these relate to software, including installation and configuration challenges, runtime dependencies, and understanding and correctly utilizing the tools available, hardware, including limitations with and access to available compute resources such as memory, CPU power and storage, and other, human-centric factors, such as the technical skills of those implementing the model. Additionally, issues spring from the complexity inherent in agent-based models. This section introduces these problems to understand how cloud services might help alleviate them.

6.2.1 Model Complexity

One challenge follows from the complexity of agent-based models. This is inherent with the nature of agent-based modelling because they are designed to capture the system complexity glossed over by statistical techniques based on representative agents or values. Model complexity has implications for the development and

[8] http://www.swarm.org/index.php/Main_Page
[9] http://cs.gmu.edu/~eclab/projects/mason/

execution environment used and the technical infrastructure deployed to simulate the model.

6.2.2 Scalability

Closely related to model complexity is the problem of scalability. Model complexity and the size of the data sets involved can result in computationally expensive simulations requiring the ability to vertically and horizontally the computational resources. The key scalability factors are computational power and memory usage. The former relates to CPU speeds and cores available. The most important aspect of memory usage, which has most bearing on the runtime behaviour of the simulation and relates to the size and growth of the simulation memory usage, is the size of the heap used by the simulation (Voss *et al*. 2010). Likely CPU and memory consumption must be factored in when deciding to use the cloud. Typically, this will involve determining the most suitable virtual machine instance type and size to meet the needs of the simulation.

6.2.3 Data Sets

There is an issue with the use of large amounts of empirical data. Often models require large data sets, which cannot be handled by local commodity resources such as PCs. This creates problems because of limits on the resources typically available to instructors in the classroom.

6.2.4 Parameter Space

Another problem stemming from the sheer size of some agent-based models – simulating a social process often incorporates millions of individual simulations – is the size of the resulting parameter space, which can lead to technical challenges. The parameter space, proportional to the number of agents in the model and the complexity of their socially salient interactions, is often huge and very detailed, consisting of multiple input parameters assuming multiple distinct values. Computationally, the exploration of this space can be very expensive and is often only feasible at scale because of the need to perform comprehensive parameter sweeps or run multiple simulations with varying or specific parameter initializations.

6.2.5 Time

The time taken to run simulations can be problematic, with some models taking [several sessions?] to fully run. This potentially presents problems if local resources used to teach a course are released after scheduled contact time for use by others. This problem does not arise in the cloud, as it is possible to leave virtual machines running indefinitely, although the cost implications of doing so must be carefully calculated beforehand.

6.3 Technical Architectures

Running social simulations, in particular agent-based models, can be very computationally expensive. The resources necessary can quickly outstrip the capabilities of even modern, powerful, desktop and laptop machines. On the other hand it is perfectly possible to run relatively advanced and complex models on cheap

commodity hardware. We believe the chief difficulty instructors face will likely stem from configuration and installation problems with the hardware at their disposal rather than insufficient access to hardware with the required technical specifications, although either is possible. Instructors can either use non-distributed local commodity resources or a (local or remote) distributed system. Distributed systems could be useful as agent models become more complex and generate computationally expensive simulations. Technical architectures capable of executing distributed simulations fall into three categories – grids, dedicated high throughput and high performance compute clusters, and clouds, although problems faced running simulation jobs on grids and distributed clusters provide a strongly motivate the latter.

6.3.1 Commodity Resources

The easiest way to run agent-based models is to use local commodity hardware, normally consisting of school desktops or user-supplied laptops. Technological advances in commodity hardware have made it possible to run relatively large-scale simulations on cheap, readily available compute resources (Voss *et la*. 2010). There are limitations in relying on institutional resources and student's laptops however. These include lack of root permissions to install required software packages, incompatible operating systems, and variability in the hardware specifications available at different institutions and with users laptops. This makes it hard to properly plan the technical aspects of a course on ABMS, as instructors have to deal with heterogeneous technologies and uncertainty over the ability of students to access the course content (particularly if the course only lasts a short period of time).

6.3.2 Distributed Systems: Compute Clusters and Grid Technologies

Deficient local resources could motivate the use of some type of distributed system. The number of agents, amount of data and size of the parameter space involved can mean that efficient processing requires the simulation to execute on a distributed system, which always introduces non-trivial configuration, deployment and administration. Particular difficulties turn on distributing the elements of a simulation over the computational resources available (Gulyas *et al*. no date given). Distributed systems include grids, dedicated clusters and clouds. Each introduces a high degree of error-prone work so their utility balances against the increased cost in complexity, configuration time and the added opportunity to introduce mistakes into the modelling and simulation process.

6.3.2.1 High Performance, High Throughput and High Availability Computing

Cluster computing refers to the use of a local collection of networked computers, where a front-end node typically acts as a load balancer and distributes work and data to back-end worker nodes. Clusters are used to provide high performance (HPC), high throughput (HTC) and high availability compute platforms. HPC is achieved by allocating compute tasks across worker nodes, to run task quicker by boosting processing power. HTC refers to performing more computations in a given time period and is achieved by increasing the number of worker nodes available. High availability computing refers to deploying redundant cluster nodes responsible

for serving applications and resources in failover scenarios resulting from systems failure, outages and other service disruptions.

6.3.2.2 Grid Computing

Grid computing extends the basic notion of a cluster to meet intensive – high performance – computational and storage requirements, with a distributed and decentralized heterogeneous compute network (in effect grids are collections of computer clusters). Grids pool distributed multivariate compute resources operated by different institutions working together, with decentralized access management mediated and controlled through middleware platforms designed to orchestrate the work of nodes located in different geographic regions with different operating systems, software stacks and hardware specifications (Feldhaus, Freitag and El Amrani 2012).

The need to arbitrate access to compute grids through complicated middleware is a big problem, whether for ABMS, teaching or other purposes, and is a strong reason to favour cloud services over grids, primarily because of restrictions on when, where and how the grid may be accessed (Feldhaus, Freitag and El Amrani 2012). Comparing clouds and grids highlights the ease of use and relative simplicity of the former for ABMS (see for example Chen *et al*. (2008), which presents a grid middleware platform designed to run Repast agent-based simulations on the grid. The complexity of this work shows how easy it is to provision cloud resources compared to grid resources).

6.3.2.3 GridABM

Often the size and complexity of agent-based models means they should execute across multiple processors. This requires code parallelization and can lead to problems if users are not proficient with advanced software engineering techniques. This is the motivation behind gridABM, a platform allowing users to easily run non-parallelized code on grids, clusters and multicore hardware.[10] GridABM is built on top of the Java programming language, the Repast modelling and execution environment and the ProActive Parallel Suite. ProActive[11] is a suite of middleware tools for the management of private and public high performance compute clouds and clusters, allowing the execution of HPC workflows and orchestrating heterogeneous resource management, application parallelization and task scheduling, and is the foundation enabling technology for gridABM. Users follow a relatively straightforward five-step recipe to run their code with gridABM. The process is designed around a set of dry schemas, which act as parallelization templates for the specific communication topology realized by a particular agent-based model. See Gulyas *et al*. (????) for further details.

6.3.3 Service-Oriented Simulation Architectures

Zhang, Coleman, Pellon and Leezer (2008) provide a service-oriented architecture designed to run multi-agent simulations across a distributed system, with a master

[10] http://gridabm.sourceforge.net/
[11] http://proactive.inria.fr/

GUI interface running on a single machine controlling the whole system. Their system aims to provide researchers with a distributed architecture that permits realistic-scale social simulations with an intuitive user interface.

6.3.4 Cloud Computing

Each of the options outlined above – commodity hardware, clusters, grids and service-oriented simulation architectures – have limitations that restrict their ability to provide instructors with reliable, easy to access, configurable, customizable, scalable and flexible compute resources. These problems strongly motivate (at least experimental) adoption of the cloud to teach ABMS. There are also positive arguments for the cloud, which we turn to after our detailed discussion of the cloud in the following section.

7 Cloud Computing

The National Institute of Standards and Technology define cloud computing as:

> *…..a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.*

Another definition from Foster *et al*. (2008) explicates cloud computing as:

> *A large-scale distributed computing paradigm that is driven by economies of scale, in which a pool of abstracted, virtualized, dynamically-scalable, managed computing power, storage, platforms, and services are delivered on demand to external customers over the Internet.*

Drawing on these and similar definitions it is possible to define the core characteristics of the cloud. Mell and Grance (2011) sum the cloud thus:

- **On-demand self-service:** Users are able to provision cloud resources, including compute power, storage, and network capacity, without personally interacting with the provider of the cloud resources.
- **Network Access:** Cloud resources are provisioned across the *public network* to run on both thick and thin clients (personal computers, mobile phones, *etc*.)
- **Resource Pooling:** the resources of cloud providers are *pooled* to create a multi-tenant access model that enables users to utilize both physical and virtual resources, which are dynamically assigned and released as demand dictates.
- **Elasticity:** Cloud resources are *elastic* because they can be dynamically scaled on the fly, allowing users to rapidly scale (in and out, vertically and horizontally) their computational infrastructure to meet changing

requirements. This provides the illusion of infinite potential compute capacity that can be instantly provisioned and released as desired.

- **Metered Service:** resource usage is controlled through a *metered pricing model* that charges users for the resources they consume, allowing both consumers and providers to monitor and control the consumption of resources.

Cloud services available across the public network include compute infrastructure, storage, user applications, networking and databases. Typically, service and deployment models are used to categorize cloud resources by the type of services offered and the manner in which they are sourced. Coarse-grained service models split the provision of services and resources into a three-component stack covering the infrastructure, development platform and application levels. Deployment models on the other hand are concerned with the nature of the cloud provider. Prevalent taxonomic variants include the public clouds of providers such as Amazon, Microsoft, and Google, and private clouds such as the University of St Andrews' research-oriented StACC Cloud. Different taxonomies do exist however, which may be of more use to those teaching ABMS. For example the cloud taxonomy given by de Oliveira, Baiao, and Mattoso (2010) looks at cloud computing specifically from an e-science perspective. This taxonomy distinguishes clouds according to privacy, pricing, architectural characteristics, technology infrastructure profiles, access, standards, task orientation and business models. Some of these categories, business models for example, map directly onto standard taxonomic categories, in this case service models, while others such as standards do not. One common and easily understood use of cloud services in the UK is the higher education sector's adoption of cloud email systems such as Gmail (e.g. the University of St Andrews' St Mail system). This is an example of a public cloud provider offering hosted private cloud services accessible only to prescribed institutional users.

7.1 Service Models

Service models are categorized by their granularity, ranging from the coarse-grained provision of low-level computational infrastructure – Infrastructure-as-a-Service – to fine grained-solutions to specific problems, such as storage-as-a-service. Infrastructure-as-a-Service denotes the wholesale provision of computing infrastructure such as servers and network capacity, for public consumption over the Internet. Storage-as-a-services relates to the online storage, management, and distribution of data in the cloud. In addition to IaaS clouds there are also clouds specifically geared towards providing development, runtime and hosting environments for application development, referred to as Platform-as-a-Service (PaaS), and cloud services offering local users access to remotely hosted software applications and services known as Software-as-a-Service (SaaS).

It is important to see each layer in the stack as building upon the level directly below it. Thus it is possible to build application development platforms on top of infrastructure offerings from Amazon EC2 for example. Likewise, SaaS can be developed and delivered on top of PaaS. For example, .NET applications can be developed on the Windows Azure platform and then delivered to end-users as SaaS.

These service models are a convenient way to view the consumption of cloud resources but it is widely noted in industry white papers, academic papers and blogs that the distinction between IaaS and PaaS is increasingly blurred (references here). For example the AWS Elastic Beanstalk allows developers to develop and deploy Java applications to EC2, and choose the degree to which they engage in low-level infrastructure management. IaaS management can be entirely automated by the Beanstalk service, which controls the provision of compute and storage facilities, or done manually as required. This is distinct from 'traditional' PaaS such as Azure, which provides a simplified development environment but does not allow the configuration and management of the underlying infrastructure services, and 'traditional' IaaS clouds such as EC2, which provides low-level infrastructure configuration but does not supply any kind of higher-level development environment.

7.1.1 Software-as-a-Service

Software-as-a-Service refers to the provision of software applications in a one-to-many relationship between providers and end-users. APIs exposed by SaaS vendors allow users to amalgamate multiple distinct software artefacts to meet their specific functional requirements. For example organizations can combine email services with software for managing the distribution and tracking of promotional email campaigns. Often SaaS aims to streamline business processes such as supply chain management, or increase collaboration amongst geographically distributed workers. Office 365[12] is an example of SaaS, as are IBM LotusLive[13] and the customer relationship management solutions available from Saleforce.com.[14] To our knowledge there are no SaaS offerings specifically designed to support ABMS, so we do not consider them any further in this report.

7.1.2 Platform-as-a-Service

Platform-as-a-Service is specifically concerned with the development and deployment of applications utilizing software and infrastructure distributed across the public network. PaaS is an abstraction that hides low-level enabling infrastructure and provides software engineers with an environment and tools support for application development. Examples of well-known PaaS offerings include Microsoft Azure[15], Google App Engine[16] (built on top of the Google Cloud Platform which we consider below), and the ActiveState Stackato[17] cloud, centred on dynamic language development (Perl, Python, Ruby, *etc.*). As with SaaS there are no ABMS-specific PaaS clouds and as their services are not aligned with the objectives of ABMS we do not consider them further in this report.

---

[12] http://www.microsoft.com/en-us/office365/online-software.aspx
[13] http://www.ibm.com/developerworks/lotus/library/lotuslive-intro/
[14] http://www.salesforce.com/uk/
[15] http://www.windowsazure.com/en-us/
[16] https://developers.google.com/appengine/
[17] http://www.activestate.com/stackato

7.1.3 Infrastructure-as-a-Service

Infrastructure-as-a-Service refers to the provision of servers, networking facilities, storage capacity, and operating systems made available over the network. The public Amazon Web Services Elastic Compute Cloud (EC2) is the preeminent exemplar of Infrastructure-as-a-Service. EC2 resources are provided in the form of virtual machines based on Amazon Machine Images (AMIs). AMIs are disk images that can be customized with an operating system and system configuration, allowing users to specify options such as CPUs, processor cores, and RAM. Bespoke software stacks are then installed to meet particular user requirements. The AMI is instantiated in the cloud at runtime, giving users access to a remote computational infrastructure. Each virtual machine is controlled through web service calls. The University of St Andrews' StACC Cloud is an example of a private IaaS cloud. This experimental research cloud allows users in the School of Computer Science to instantiate a variety of machine images with different operating systems and custom configurations. A fuller and more detailed description of the infrastructure services provided by EC2, and a comparison with the Google Compute Engine, is given below.

7.1.4 Eduserve

Eduserve is a public IT and cloud services provider serving organizations in the UK.
[Introduce/discuss Eduserve here?]

7.1.5 Other Cloud Services (Storage-as-a-Service, Database-as-a-Service, etc.)

In addition to the ubiquitous service models above other cloud services exist that are given varying classifications by different authors. For example de Oliveira *et al*. consider Storage-as-a-Service and Database-as-a-Service as particularly important to distinguish within the context of running scientific experiments in the cloud, given the central importance and value of the novel data generated by scientific experiments. As with other more specialised and narrowly focused cloud services catering for specific user needs, this report does not investigate the gamut of service permutations available in the nascent though fast developing cloud marketplace, because they are likely to hold little current interest for those teaching ABMS.

7.2 Deployment Models

EC2 and StACC illustrate different cloud deployment models. The chief distinction is between public and private cloud deployments. AWS *public cloud* resources are available over the public Internet and can be provisioned by users on the basis of a self-service, utility-style pricing model (users pay for the computational resources they consume). By contrast StACC is a *private cloud* that provides services only to internal users at St Andrews. The chief alternative to these deployment models is the *hybrid cloud*, which utilizes both public and private clouds. With hybrid clouds, researchers can, for example, work with private cloud resources and then 'spill' heavy workloads onto EC2 as required, in a process known as *cloud bursting*. Finally, there are *community clouds*. Community clouds provide access to cloud resources to the specific communities of interest responsible for their implementation and management, so are clouds used and shared by communities of interested parties. The Open Cirrus Cloud Testbed is an example of collaboration between cloud users

(Campbell *et al*. 2009). Open Cirrus is operated by companies such as HP, Intel and Yahoo, and aims to enable research into large-scale IT services, as well as providing a platform for the exchange of best practice and the dissemination of knowledge generated through collaborative research. Potential issues faced when operating community clouds include access security, service levels, resource availability, data storage and compliance (*Ibid*.). Whilst there is potential for the development of community clouds dedicated to or connected with teaching and research into ABMS, community clouds are not relevant to our present concern of establishing the potential benefits and challenges posed by integration cloud services into ABMS teaching strategies. Public, private and hybrid clouds are however relevant and are considered next.

7.2.1 Public Clouds

Both private and public sector organisations operate public clouds. AWS, which supplies cloud resources through EC2 and related services, is the most dominant public infrastructure services cloud in the marketplace today. EduRoam[18] is an example of a public cloud operated by a federation of research and educational establishments that span 54 countries across Europe, North and South America, Africa and Asia, with the goal of facilitating some of the research and communication computing requirements of member organisations. One key objective driving the development and gradual adoption of public cloud computational infrastructure is the ability to hide the complexity associated with managing and administrating complicated compute resources whilst simultaneously exposing self-service functionality to end-users. Resource multiplexing and multi-tenant architectures enable higher resource utilization than traditional data centres built on top of a single-tenant server infrastructure. Public clouds are typically accessed via metered self-service without interaction between the cloud provider and those accessing their services. With respect to the service models outlined above public clouds can provide IaaS, PaaS and SaaS resources.

7.2.2 Private Clouds

Private clouds provide their organisations with internal access to their own dedicated cloud resources. In this way they imitate the functionality of public clouds but restrict access to defined corporate or institutional users. Two usage models support private clouds (reference here). With in-house or internal private clouds operators own the physical data centre resources that supports the cloud and are fully responsible for configuration, monitoring, maintenance, upgrades, administration, *etc*. Alternatively, hosted or external private clouds are operated and maintained by third-party vendors for the sole use of a particular organisation or client. Typically organisations operate private clouds when they have either specific resource demands that necessitate close control of physical resources, such as a software testing house, or they have particular and stringent legal data storage requirements (under the Data Protection Act for example).

[18] http://www.eduroam.org/

7.2.2.1 The StACC Cloud and similar University Private Clouds

The School of Computer Science at the University of St Andrews operates a private cloud called StACC[19] (St Andrews Academic Compute Cloud), which runs the Eucalyptus management framework and gives researchers access to up to 64 virtual machines running Linux. Other UK universities host similar private research clouds including Leeds and Oxford [check this!]
[Expand StACC discussion?]

7.2.3 Hybrid Clouds

Hybrid clouds make use of both public and private cloud resources. They are particularly suited to running jobs with dynamic workloads where resource consumption can spike, requiring access to extra resources, which may not be available locally. The process of offloading the execution of a software application operating under excessive load spikes onto a cloud is known as cloud bursting. Cloud bursting is not the only or even primary use of hybrid clouds. They are also used as failover platforms. For example, if a simulation run is executing on a private cloud and those resources fail the simulation can be setup to automatically provision and configure public cloud resources to facilitate the simulation in place of the private resources. Of course many factors impact the effectiveness of this strategy, such as the latency between the initialisation of the public resources and the establishment of a fully operational system.

7.2.3.1 Cloud Bursting

Cloud bursting is the process of spilling spikes in computational load onto the cloud when local resources are insufficient to meet demand. Zhang, Jiang, Yoshihira, Chen and Saxena (2009) develop a "two zone architecture" to conceptualize cloud bursting. This consists of a base load and a trespassing load, and this way of thinking is helpful in explicating the notion of a hybrid cloud and the concept of cloud bursting. Standard or normal work done by an application is termed the *base load*. The *trespassing load* represents non-standard (occurring less than 5% of application runtime e.g.), transient spikes in activity that increases short-term load on the system and requires extra resources. This is one way to use hybrid clouds – leveraging on-demand elastic cloud resources to meet trespassing load by bursting the computation onto the cloud – in addition to the failover strategies introduced above.

7.3 Virtualization and Middleware Platforms

This section introduces two core technologies that enable the cloud – middleware platforms and virtualization. Access to cloud resources is mediated and controlled by cloud middleware platforms – referred to as management frameworks – although how this is done is not specified in the NIST definition [API comment here]. In practice management frameworks typically leverage virtualization and web services to realize an API providing on-demand, multi-tenant, elastic computational environments, providing a programmatic, command line or GUI interface to the cloud. AWS for example exposes SOAP and RESTful APIs, a command line tools package and a GUI management tool. Although standardized APIs have not been

[19] http://www.cs.st-andrews.ac.uk/stacc

universally adopted by industry, the EC2 API acts as a *de facto* industry standard for cloud vendors and software houses, and is the closest approximation to codified cloud technology standards currently in production (Feldhaus, Freitag and El Amrani 2012).[20]

Virtualization allows the deployment of multiple logical servers onto a single physical host server, and allows cloud users to instantiate virtual server instances on-demand. To control this process virtual server instances are installed onto a hypervisor that can manage suites of guest virtual machines running on their host machine. Hypervisors are categorized as either *bare metal* (type 1) or *hosted* (type 2). Bare metal hypervisors are installed directly onto the host server. Guest operating systems (Linux, Windows, Solaris e.g.) are then installed and run on top of the hypervisor. Hosted hypervisors are installed and run on top of a base operating system. In either case physical resources are abstracted from end-users and application functionality, permitting hardware resources to be pooled and distributed on-demand. A cloud management framework orchestrates the provision and configuration of the virtual machines exposed by the hypervisor. The interaction between end-users and the management framework forms the cloud front-end. The framework exposes an interface permitting users to configure and manipulate virtual machines using web service calls, and to create and assign security groups that mediate and control access to the virtual machine instances. Typically machine images are held in a repository and instantiated at runtime by individual virtual machine instances, and act as configurable templates that are customized according to the needs of users and the particular demands of the hypervisor.

Virtualization software is aimed at individual, institutional or corporate use. The former allows users to configure their own virtual machine images and launch them on their own local resources. They are also useful for booting disk images that have already been run in the cloud and have generated data that users wish to take away and explore further. The latter include virtualization products designed to support data centres running multiple virtual machines, and enable the operation of cloud infrastructures. Major virtualization platforms currently in use include VirtualBox and the many virtualization products owned by VMware. VirtualBox[21] is an open source, free to use virtualization application developed and maintained by Oracle. It is installable on Linux, Windows, Solaris and Mac hosts, and supports the virtualization of a wide range of operating systems including multiple flavours of Linux, Windows (XP, Vista, 7, Server 2003, *etc*.), (Open) Solaris, and OpenBSD. VMware[22] is a company specialising in virtualization products for personal, academic and enterprise use. For example VMware Fusion allows Macs to run Windows, Linux and

[20] The Open Cloud Computing Interface (OCCI) is currently attempting to provide standardized access to IaaS resources, whilst the development of software libraries for use with different hypervisors, and libraries used to replace the exposed heterogeneous resources of different clouds with an homogeneous interface, are also under active development (*Ibid*.).

[21] https://www.virtualbox.org/

[22] http://www.vmware.com/uk/

other operating system. Other products include free to use desktop virtualization platforms such as VMware Player, supporting 1-2 virtual machines per host, free to use multi-server virtualization platforms such as VMware vSphere Hypervisor (ESXi), which supports less than 10 virtual machines per host, and other paid for platforms such as VMware vSphere that support greater than 10 guest machines per host, as well as providing centralized management services and other features not available with free editions.

It is important that instructors looking to use the cloud understand how different virtualization technologies can affect and restrict their activities. This is because each cloud makes use of different virtualization platforms that in turn dictate the type of virtual machine image that can run on that cloud. For example EC2 uses the Citrix Xen hypervisor, a type 1 (bare metal) hypervisor. This means that virtual machines images booted on EC2 must run a Xen kernel. If the Xen kernel is not installed users must do so manually, involving a complex and error prone configuration process.

This means the technologies used by mainstream cloud vendors restrict the types of virtual machine image files supported by their infrastructure. If the image is not in a format supported by the cloud vendor they must be converted to a compatible format. For example EC2 only allows users to import virtual machine image files (VMDK and VHD file formats) that are created using the following platforms:

- VMware ESX (VMDK files)
- Citrix Xen (VHD files)
- Microsoft Hyper-V (VHD files - for use with Windows Server 2003 R2 and 2008 R1 and R2 only)

In a similar fashion custom VMs that have been imported into EC2 (or instantiated from a pre-existing AWS template) can only be exported into these same file formats (as well as the VMware ESX OVA format). Thus if a VM is created with VirtualBox and saved as a VMDK file, it must be converted to the appropriate file type before the EC2 VM import/export command line tools will work properly, and the same is true in the other direction – VMs exported from EC2 must be converted to the format that your local virtualization infrastructure supports. This problem is not lethal but is unfortunate because it introduces a significant level of complexity into the configuration process – a misfortune highlighted by the supposed ease with which cloud resources can be provisioned relative to clusters and grids, which is a central motivation for adopting the cloud. The solution is to use conversion software such as VMware converter but this is non-trivial and represents the type of low-level configuration work that many users are not comfortable with without a background in computer science or other significant experience.

For this reason we do not recommend that instructors wishing to use the cloud do so by manually configuring their own virtual machine images locally before exporting them to the cloud. Doing so potentially involves protracted configuration, with all of the errors, oversights and frustration this can entail, and can also restrict the portability of the image between clouds. Instead of comparing which virtualization

platform disk image file types are compatible with which cloud – i.e. which VM disk image can be converted to run on a particular cloud – instructors considering bundling custom simulation disk images are advised instead to either manually configure a template image that is already hosted by their chosen cloud vendor, or, preferably, to run a script that automatically configures an image for their chosen purpose. This is known as contextualization and is considered below.

7.4 Cloud Economics

This section introduces financial considerations instructors should consider when evaluating the cloud.[23] The basic cost argument for public clouds is simple – on-demand self-service allows users to access only those compute resources they require, for the specific time period they are needed. Users only pay for the resources consumed and cease to pay when they are released, thus lowering total technology spend. Access to powerful resources is available in the short-run at very low prices – 'operational' costs – relative to the 'capital' costs associated with the procurement, installation, configuration and maintenance of new physical servers and related hardware.

Unfortunately comparing cloud providers on cost is not easy. Users need to estimate fine-grained usage patterns, estimating for example outbound data transfer in advance and use this to calculate likely spend given a vendor's prices.[24] Some vendors, AWS for example, include cost calculators tools on their web pages, but these are not easy to use. The AWS tool is complex, detailed, and tailored towards calculating monthly usage, which is unhelpful in procuring short-term resources for a course on ABMS.

7.4.1 Cloud Costs Incurred during a typical ABMS Study Programme

Instructors can expect to directly incur costs for using cloud services across three separate components (JISC Cloud Costs Report):

1. Ingress data transfer costs
2. Infrastructure Costs:
    a. CPU instance hours
    b. Data transfer
    c. Data storage
    d. [Other costs]
3. Egress data transfer costs

In addition other costs are likely to be incurred such as:

1. Software licensing
2. [Other costs here]

[23] See [Jisc Cloud Report] for more information on the costs of cloud computing.
[24] See for example http://calculator.s3.amazonaws.com/calc5.html

Total costs are directly dependent on usage over the lifetime of a course. If students boot virtual machines and do not terminate them costs are incurred until they are shut down. On the other hand using a public cloud to demonstrate a modelling platform or execute a simulation only invites small usage costs even if the demonstration resources are instantiated over several days and not torn down outwith contact hours.

Any price information contained in this report will be quickly superseded in the market so we do not provide detailed cost profiles for different cloud providers. JISC notes that cost per CPU core-hour range on average between approximately 7p and 10p, with costs reaching as high as 19p for larger machine instance types. It is possible however that data storage fees incurred over a typical ABMS course are likely minimal, if any. Similarly, costs for data transfer should be low. Many vendors no longer charge for ingress data transfer, while egress costs are low. JISC estimates that outbound transfer costs for 1TB of data range between £0-£200, but it is highly unlikely that participants will generate 1TB of data unless the focus is on HPC simulation or other data intensive simulations. Models developed using platforms such Repast Simphony are likely to generate only small amounts of data and so costs per head are likely to remain negligible.

7.4.2 Visible Costs

Total cost will also be far more visible to instructors, students and administrators (JISC Cloud Cost Report). Traditional academic in-house technology costs are hidden from end-users – academics and students just turn up and use the available resources. In contrast, cloud-metering services make the costs associated with specific uses of specific technologies at specific times for specific purposes explicit for both users and administrators.

7.4.3 Payment Models and Charging Models

Payment models vary between vendors, users typically incurring monthly charges paid using a debit or credit card. Usage is generally charged by clock or CPU hours consumed, hence it is also important to understand which charging model a vendor employs and how their billing model operates (JISC Cloud Cost Report). To understand the full cost behaviour of their study programmes instructors should also understand how to cover their cloud expenses – does the use of school credit cards require prior approval and for what amounts, for example, and who must authorise this expenditure?

7.4.4 Software Licensing

Software running in the cloud is subject to the same licensing restrictions that apply to software not running in the cloud. For some applications these are minimal. Multiuser licences for remote access software such as Real VNC Server can be purchased for only [a price]. Other types of software incur higher licensing costs (Microsoft Windows for example).

7.4.5 Recovering Costs

One way to obviate cloud costs is to incorporate at least some of the additional costs in increased course fees, with fees structured to explicitly take into account the costs of running ABMS jobs in the cloud, licensing and storage.

7.4.6 Comparing Cloud Providers on Cost

PlanForCloud.com[25] is a web service that allows users to generate cost forecasts based on their requirements and then compare costs from different cloud service providers. It provides cost breakdowns for servers, databases, storage and data transfer, and allows users to exactly specify their requirements across these resource types. Users can also factor usage patterns and variations into their cost forecasts. At present PlanForCloud.com only supports cost comparisons for Amazon Web Services, Microsoft Windows Azure, Google Compute Cloud and Rackspace. How useful this service will be for teachers is hard to tell, as the emphasis seems to be on long-term enterprise deployments, but the site does provide convenient access to bespoke cost comparison data for four of the major public cloud providers.

7.5 Comparing IaaS Clouds – AWS and Google Compute Engine

To give readers some idea of the different infrastructure services available this section compares the market-dominant EC2 with the beta Google Compute Engine (GCE). AWS is such a dominant role in the current public cloud ecosystem that it is highly likely that if instructors have made the decision to use the cloud they will initially consider AWS to the exclusion of other platforms. On the other hand Google's Compute Engine is still in beta but is set to become a serious IaaS player. We described the services available from EC2 and GCE that are most relevant and for ABMS in general and teaching ABMS in particular.

7.5.1 Amazon Web Services

AWS offers a number of infrastructure services. These include the Elastic Compute Cloud (EC2), Simple Storage Service (S3), SimpleDB (SDB), and Simple Queue Service (SQS). There are many more services available and the reader is directed to the AWS web pages for further information.[26] In brief:

- **Elastic Compute Cloud (EC2):** The core AWS infrastructure service allowing users to provision configurable virtual server instances on-demand. Amazon Machine Images (AMIs) act as templates that are instantiated by one of Amazon's standard instance types, which specify the hardware specification in terms of CPU speeds, cores and RAM
- **Simple Storage Service (S3):** S3 is a non-relational storage service giving users quick, dependable, convenient storage and retrieval of data in the

[25] http://www.planforcloud.com/

[26] See Rittinghouse and Ransome (2010) for an easy to follow introduction to cloud computing and comparison of AWS and Rackspace IaaS services.

Amazon. Template AMIs and user-configured virtual machine images are stored on S3 for use with EC2.

- **SimpleDB (SDB):** SDB facilitates the storage, analysis and querying of structured data held in the S3.
- **Simple Queue Service (SQS):** SQS allows messages that are passed between compute nodes to be queued in the cloud.

We concentrate on EC2, as it is the service of most potential use to instructors.

7.5.1.1 EC2 Instance Specifications

[Specifications here]

Some of the major advantages of using EC2 to deliver ABMS study programmes include:

- Simple access to and configuration of compute infrastructure through a Web services interface. This allows instructors to easily provision their exact infrastructure requirements through detailing hardware specification such as processor cores and main memory, while fine-grained/low level configuration and control allows simulation and teaching environments to be customised to meet a very wide range of ABMS teaching scenarios.
- Lower infrastructure procurement times – virtual servers are bootable in minutes where physical servers are procured through lengthy processes (e.g. getting your department to pay for new hardware, configure, install, and maintain it).
- Massive and rapid horizontal and vertical scalability. Horizontal scalability refers to adding more of the same type of capacity i.e. more servers of the same hardware specification. Vertical scalability refers to the addition of servers with higher specifications. This allows jobs requiring HPC or HTC resources to be easily run at short notice with minimal configuration.
- Expenditure on infrastructure capacity for specific simulation or teaching projects can be planned because of the on-demand self-service pricing model.
- Access to powerful resources without heavy, up-front, long-term capital expenditure. Costs are only incurred when resources are used for a specific purpose and can be planned for in advance.
- Public clouds, especially clouds as easily accessible as EC2, give instructors great flexibility to experiment with new simulation types, teaching strategies and in general the ability to trial new ideas and approaches to (teaching) ABMS that are not necessarily possible with in-house infrastructure.
- Geographical flexibility/mobility. EC2 allows instructors to independently design and plan their ABMS teaching programmes without worrying about the specific resource stacks and configurations available to them through their institutional hosts. Instructors have repeat access to the resource stacks they require regardless of where they are delivering the programme.

7.5.2 Running a Simulation on EC2

So that readers have some idea of the steps required to run a simulation in the cloud we outline the major steps required. After signing-up to AWS using their normal Amazon account to access EC2:

I. Users must choose either a preconfigured Amazon Machine Image based on a template or design their own machine image to satisfy their requirements. The custom machine image contains their chosen operating system, applications, data, libraries, compilers, runtime environments, remote access software, and in general anything else required by the user, including full user-supplied configuration.
II. The Simple Storage Service is used to store machine images after they are exported to the cloud, ready for deployment at runtime, so if users are using their own custom AMIs they must upload them to S3.
III. Network access and security protocols must be configured by the user to control access to, and use of, their virtual resources, including configuring firewalls, user permissions and establishing group policies.
IV. Users must choose the instance types they will use to instantiate their machine images. An AWS instance type refers to the specification of the virtual machine used to boot the machine image. It is here that users specify the processing power, memory and configuration of their virtual machine instances.
V. Virtual resources are instantiated, controlled, monitored and terminated via web service calls using either the AWS command line API or through GUI front-ends exposed by AWS or other third party cloud monitoring platforms.

This list is not comprehensive or detailed, and does not aim at step-by-step instructions. Rather it paints a rough picture of how to run an agent-based model on EC2.

7.5.3 Google Compute Platform

Unlike the multifarious AWS product suite Google's nascent Cloud Platform consists of only a small number of services. These are the established App Engine, delivering Platform-as-a-Service, the beta Google Compute Engine, offering infrastructure services, Google Cloud Storage, providing large-scale data storage, and Google BigQuery, offering big data SQL-based business analytics services. In addition Google also offers machine learning and prediction services through Google Prediction API and translation services in the guise of the Google Translation API.

7.5.3.1 GCE Instance Specifications

At the time of writing Google Compute Engine (GCE), although in its infancy, appears set to become a major rival to Amazon Web Services (tech blog/news reference here). GCE offers 4 virtual machine types with 1, 2, 4 and 8 processors cores, with

main memory ranging from 3.75GB through 7, 15 and 30GB and local storage at 420GB, 870, 1770 and 2 x 1770GB per virtual machine.

Both AWS and GCE offer infrastructure services that are broadly comparable. In terms of cost it is necessary to compare the costs of virtual machine deployment, ingress and egress network traffic, persistent disk data storage, and static IP address assignation (ephemeral/transient IP addressing is free from both providers). However medium- to long-term fluctuations in pricing structures means there is little use in providing detailed comparisons and tables of price data.

8 Using the Cloud to Teach ABMS

8.1 Arguments for the Cloud

A number of arguments motivate the move to cloud, including limitations and problems with local technology stacks, flexibility and configuration management.

8.1.1 Configuration Management

One motivation for the cloud follows from problems with hardware and software. We term this configuration management. Hardware issues centre on access to resources and technical specifications. Issues with software tend to focus on installation problems, licensing, platform support, and runtime dependencies. Particular problems faced depend on how the course is delivered – factors affecting access to hard and soft computer resources include whether or not the course is being delivered at a home or guest institution, if the course is being delivered remotely or locally, if students are using their own (laptop) computers or institutional desktops, and the cost associated with acquiring new resources such as software licenses.

8.1.1.1 Hardware Configuration

Access to the necessary hardware is typically through the institution hosting the study programme, including basic access to desktops, servers and networks. There is also scope and often expectation for participants to bring the their own laptops. Both options have potential difficulties such as restrictions on:

- Memory, processor speeds and cores, local storage, and graphics processing capabilities
- Access credentials, user accounts, group policies and security protocols
- Install and configuration permissions (root access *e.g.*)
- Network access – Ethernet and Wi-Fi
- Network quality – bandwidth, download/upload, import/export speeds
- Storage requirements – especially with high-volume results data generation
- Display requirements – GPU, *etc*.

In the context of a small-scale short-course on ABMS running small-scale simulations developed on the smaller-scale prototyping environments such as Repast Simphony, local desktop and laptops should be sufficient. Problems arise in configuration, as it is unlikely that user or school machines will be installed with the required packages and dependencies, making it necessary to run install scripts under root permissions to establish a suitable runtime environment.

8.1.1.2 Software Configuration

Potential difficulties with software include:

- Licensing – single and multi-user
- Operating system support
- Runtime dependencies:
    - Libraries
    - Compilers
    - Version support
    - Language support
    - Drivers
    - Configuration problems resulting from the use of heterogeneous software stacks

These issues also fall under the broad aegis of configuration management and can require teachers and students to invest significant time and effort in preparation for even basic simulation runs that could be better spent pursuing the course objectives or other work.

The use of locally fabricated virtual machine images containing the software packages and dependencies required to launch and execute an agent-based modelling environment, the use of contextualisation scripts, or even manual configuration of virtual machines running in the cloud can avoid problems such as operating system support or the lack of root install permissions that arise when installing software on school resources or student laptops. We describe these approaches to using the cloud below.

Adopting the cloud can help to meet these configuration challenges. Infrastructure services provide quick access to massively scalable compute resources. This allows instructors to provision exactly those resources they need for a specific purpose, with deployment times measurable in minutes and users having the necessary install permissions. For example, below we demonstrate the use of EC2 to launch a virtual machine instance bundled with the Repast modelling environment and associated data. We specified processor speeds, cores, and main memory, mirroring the process instructors could follow to provision cloud services and avoid problems associated with configuration management.

8.1.2 Flexibility

The cloud gives teachers access to a powerful, configurable and flexible technology stack that is independent of the institution hosting their course and does not rely on students supplying their own laptop computers for use during class. Of course a

computer of some type is required to access the cloud but any machine with a GUI and Internet connection will do (including mobile). This gives instructors a degree of flexibility when planning their study programmes. Accessing cloud resources only requires access to a good network connection and commodity hardware. So as long as a course is delivered at an institution with an Internet connection (a given) instructors are able to access a heterogeneous, powerful and hugely scalable technology stack, giving them extensive options to tailor the exact resources they need for their specific purposes before releasing them after they are no longer needed.

8.1.3 High Performance Computing

The cloud gives users access to compute resources at a scale unparalleled in traditional higher education teaching and research. Instead of relying on complex grid resource management frameworks, access to supercomputers or insufficiently powerful local clusters and desktops, instructors can use clouds such as EC2 to easily scale their compute resources to run high performance jobs on-demand. This could involve conducting an entire simulation run as an HPC job in the cloud, hybrid cloud setups using school and public cloud services or bursting HPC jobs onto the cloud.

8.1.4 High Throughput Computing

HTC workflows use batch processing to maximize the number of operations performed in a given period – floating-point operations per second e.g.. Batch processing in the cloud requires a task management system to schedule, setup, run and teardown the necessary resources. One option is to use a continuous integration build server such as Jenkins. Jenkins is primarily intended to provide a continuous integration platform to automate the build, integration and testing of software. As such they expose reliable, dependable and extremely effective scheduling and task management functionality that can manage batch simulations in the cloud.

8.1.5 Remote Access Software

Running batch or interactive tasks in the cloud requires remote control software to support graphical and command line programs. Depending on the platform native options include SSH, a Unix command line utility allowing remote access across the network, and remote desktop applications enabling local invocation and display of remote applications such as the open source solutions based on Virtual Network Computing (VNC) and proprietary platforms such as Netop, and (Netop Report).

8.2 Choosing which Cloud Technologies to Use

This section looks at factors instructors should keep in mind when evaluating cloud services and vendors. The initial decision is to determine the appropriate service and deployment models, before considering particular services. Resources already available will shape these decisions, but cloud services can be difficult to properly evaluate so the choice is not necessarily straightforward.

Instructors should have some idea of the variance in cloud performance, factors responsible for the variation and the metrics used to test, benchmark and evaluate different service types from different vendors.

Detailed performance metrics are not very useful for ABMS teachers because performance variations are minimal relative to typical ABMS use cases.

Lack of information from by cloud vendors is one reason for this. The other problem is that although systematic studies exist that evaluate different cloud services across a range of metrics, and some compare performance variability of specific clouds over time, these studies tend to drill down. Iosup, Yigitbasi and Epema (2011) for example investigate the performance variability of some cloud services provided by Google and Amazon.

8.3 Evaluating Cloud Services

Typical evaluation metrics include availability, throughput, utilization, and performance, although how they are defined varies. Hyperic,[27] for example, is a tool designed to monitor system and application performance of software running in the cloud and on other virtualization platform. The performance information Hyperic collects is helpful in introducing the types of performance evaluation necessary when making non-trivial or long run use of the cloud. Availability can be defined in terms of whether or not a cloud service can be contacted and is able to serve requests. Throughput is measured in different ways depending on whether the service measured is a web or application server, or a database application. For web and application servers' throughput is measured in terms of bytes or requests received and served across a designated time period. Database application throughput, again measured over a designated time frame, is measured in terms of open connections and requests processed. The approach taken to measure resource utilization also varies and is dependent on the individual platforms, resources, services, and servers comprising the application under evaluation. Hyperic gives users the possibility to generate an overview of the capacity available across an application as a whole and to drill down into specific elements to examine resource underutilization and application performance bottlenecks. Finally, performance metrics are typically formulated in either temporal units (time taken to perform operation x) or integer values (i.e. message queue lengths).

It seems reasonably clear that these types of metric are of less interest and use to a teacher planning an ABMS study programme than the analyst evaluating an enterprise deployment, but it may transpire that teachers wish to compare different cloud options against each other and it is across these type of quality of service metrics that such a comparison can be made. We envision that the most useful comparisons for teachers though will be on the types of infrastructure services available, the type of virtualisation platform used – as this determines the way bespoke virtual machine images can be deployed to the cloud – how access to the cloud infrastructure is achieved, and, perhaps most importantly, cost. The

[27] http://www.hyperic.com/

8.3.1 Service and Deployment Models

In terms of service models, infrastructure cloud services are of primary interest for teaching ABMS. The Model Exploration Service outlined in [a section above] is an instance of Simulation-software-as-a-Service, but at present such projects are rare and lack universal applicability for different teachers in different contexts, aiming at different objectives. There are no SaaS vendors who deliver simulation-centric software as a commercial service. Similarly, Platform-as-a-Service vendors are yet to host dedicated agent-based modelling and execution environments on their cloud stacks. Were SaaS vendors to start delivering simulation software this could greatly simplify the delivery of ABMS study programmes, provided there was sufficient configuration flexibility and maximum leeway for instructors to control input parameters and tailor the software for their purposes. Likewise, the provision of dedicated PaaS modelling and execution platforms could have beneficial ramifications for students and teachers of ABMS by allowing them easy access to third-party hosted development resources. Again there are no PaaS offerings in the market.

Looking forward it is possible that both SaaS and PaaS simulation services could help provide the backbone of an e-learning agent-based modelling and simulation cloud ecosystem but this is speculative (see discuss this below). Presently it is reasonable to surmise that if teachers choose to deploy cloud resources they will use IaaS cloud services as these provide low-level access to and configuration of core computing infrastructure resources.

Considering deployment models it seems that public, private and hybrid clouds all have the potential to help AMBS teachers. Here decisions are likely to turn on what access a teacher has to what types of resources. For example, a simulation course hosted by the University of St Andrews could use the StACC cloud, although access to other private clouds hosted by other universities or other institutions is easy enough subject to network connectivity and agreements between institutions and instructors. If simulation runs executing on StACC were then farmed-out to EC2 or the Google Compute Engine the course would in effect be leveraging hybrid cloud services - the private StACC cloud and the public AWS EC2. Alternatively the entire course could be run on EC2, or another public infrastructure cloud, making use of only public cloud services. Decisions here are likely to be framed by what resources are available locally, where the course is being delivered, the aims and content of the course, and other factors such as cost and a particular cloud's virtualization environment. Other considerations, normally key when evaluating cloud services and deployment models, such as reliability, quality of service and service level agreements, could also be relevant, but they have less influence within the context of teaching agent-based simulation than if configuring cloud services for commercial deployment, because the performance characteristics of IaaS clouds are all broadly similar for the limited set of short-run tasks required during a course on ABMS. It is reasonable to conclude therefore that without the development of substantial

private cloud capacity it is likely the majority of ABMS teachers using cloud technologies will access public cloud services if they choose to use them.

The arguments for the cloud outlined above are based on limitations and problems that teachers are likely to face while delivering a course on agent-based modelling. The next section looks at six ways teachers can use the cloud to circumvent these problems.

9 How Teachers can make best use of the Cloud

This section discusses the different options available for running agent-based simulations in the cloud. It runs through each option, including a description of the use case and its associated problems. Running simple simulation models in the cloud should be relatively straightforward given access to the appropriate local resources – namely a client machine (desktop, laptop), network connection and payment method –but there are potential difficulties. Given the restrictions imposed by the specific virtualization platform in use by a particular cloud vendor that we discuss above, we identify two central use cases for the cloud that readers are likely to adopt, which involve either manually or automatically configuring a template VM that is already running in the cloud. We also discuss exporting virtual machines from the cloud, bursting simulations onto the cloud, and a specialised offering know as the Model Exploration Service, which allows batch simulation jobs to be run on EC2 with minimal configuration effort.

9.1 Booting a hosted template image in the cloud with manual configuration

The most straightforward way to use cloud infrastructure services is to manually boot and configure a template virtual machine image that is hosted by the cloud services provider. The process of installing software packages and configuring a template virtual machine for a specific purpose is known as contextualization. Contextualization can be done manually or via an automated script, and the main effort necessary with using template images is the manual contextualization process.

A GUI such as the AWS Management Console or a command line API is used to launch, control and configure each template. For example users could boot a template Ubuntu EC2 instance with sufficient memory and processor power to run their simulations before installing and configuring the necessary software packages. Using a Linux package manager such as apt (Ubuntu) or yum (CentOS) can make the installation of packages relatively straightforward although configuration can be more difficult. The packages and dependencies required will vary between uses but would probably include a modelling and execution platform such as Repast Simphony with an associated development environment (Eclipse is bundled with Repast Simphony). The modelling platform will require a runtime environment such as the Java runtime OpenJDK and any runtime dependencies, for example compilers, libraries, and pseudo-random number generators also need to be installed and configured. If using Linux it can be reasonably assumed that the required packages are either already installed or are easily installable using the native package manager. The GNU Compiler Collection, which includes the GNU C, C++, and Java compilers, is easily installed on Ubuntu via the Build Essentials meta-package. The

picture is complicated if running proprietary software or if the operating system does not have basic packages as standard (e.g. Microsoft Windows does not have a native Python installation). In this case manual search and installation is necessary. Executing a simulation and viewing the results requires a graphical interface (in place of SSH from the command line). This requires remote access software such as VNC Server to be installed and configured on the virtual machine. Finally any other packages, a database server such as MySQL for example, must be manually installed and configured.

The point here is that although manual contextualization can be relatively straightforward – although this is by no means the case as installation and configuration can be highly non-trivial and require significant effort – it in large part obviates the ease with which cloud resources can be accessed, increasing the time, complexity and effort for students and teachers using the cloud. Furthermore persisting the VM incurs additional costs or manual re-contextualization between sessions, unless the VM is kept running, which is financially prohibitive and likely undesirable. There are two ways to avoid this effort – the first is to run a contextualization script that performs the installation and configuration effort on a template VM running in the cloud. The second is to manually bundle and configure a custom virtual machine using a local virtualization environment such as VirtualBox before uploading it to the cloud.

9.2 Booting a hosted template image in the cloud and running a contextualization script

Using a contextualisation script to configure a VM has a number of advantages in addition to the reduced effort described above. Firstly the contextualised images will always be up to date, provided the software repositories the script accesses contain up to date binaries. Having to manually configure and compile the necessary software to obtain up to date binaries would of course obviate the benefits of the contextualisation script. Fully up to date software packages are less likely if the virtual machine is manually configured, updated and actively managed. There is also less effort involved with switching cloud infrastructures because the script can be run on the template virtual machines native to a particular infrastructure. As different clouds use different virtualization platforms, different clouds run different types of (possibly incompatible) virtual machines. This reduces the portability of VMs bundled for a particular cloud and restricts the ease with which users can switch clouds. Contextualization scripts alleviate this difficulty. Using a repository to store contextualization scripts also allows specific teaching materials to be version controlled. The development of version-controlled repositories containing scripts tailored to specific simulation jobs has the potential to greatly reduce the time and effort instructors must invest before leveraging the benefits of the cloud.

9.3 Manually configuring a local image and importing it to the cloud

A third use case is to locally bundle virtual machine images containing everything necessary to develop and execute agent-based models and importing them to the cloud. After bundling an appropriate machine image, users upload it to the cloud using a suitable API. Locally bundled images are similar to those described above – a base operating system, software toolkit and modelling environment, user- or

simulation-specific data and any other runtime dependencies. After the VM is imported to the cloud it is booted on an appropriate machine instance.

Unfortunately this use case is highly problematic because of the virtualization restrictions outlined above.

9.4 Exporting virtual machine images from the cloud

The third way users are likely to use the cloud is to export their machine images, simulation results and data from the cloud to their local infrastructure. It is important that after students have run their models in the cloud during course contact time they are able to export their VMs from the cloud to boot on their local virtualization platforms. In general this process allows people to port their results between clouds and other infrastructure platforms, allows teachers to build persistent educational resources based on simulation results generated in the cloud.

9.5 Cloud Bursting

Cloud bursting is another potentially useful application of cloud resources if executing the simulation realizes high performance load spikes that are easily transferrable to the cloud. This will depend on the simulation at hand. For example simulation runs using the Repast for High Performance Computing modelling and execution platform could require access to resources beyond those otherwise available. It is perhaps unlikely however that readers of this report will choose to run such simulations given that we focus on small-scale teaching contexts which do not require powerful compute resources. Bursting simulation runs onto the cloud is though a live option and could be very useful in particular circumstances.

9.6 Virtual Machine Image Contextualization Repositories

As demonstrated below running simulations in the cloud is relatively straightforward once a suitable virtual machine image has either been created in the cloud or exported to the cloud. In the case of EC2 this involves bundling a machine image using local virtualization technologies, uploading the image to the AWS Simple Storage Service (S3) using command line tools, before instantiation using one of the available AWS instance types at runtime.

This suggests that one way to further uptake and ease the use of the cloud is to develop repositories or libraries of contextualization scripts which can be rapidly and easily deployed by teachers in line with their particular teaching requirements, potentially alleviating (at least part of) the bundling and configuration effort required to produce a machine image for upload and instantiation on to public, private or hybrid clouds, or the manual configuration necessary when booting hosted template images. The repository could be hosted by an online service such as Bitbucket [28] or privately by communities of interested parties. Students and instructors could reliably select and launch the resources required without having to

[28] https://bitbucket.org/

undergo complex and error-prone configuration prior to or during class time. Users can quickly boot customised, configured virtual machine instances, greatly reducing the time and effort needed to build a suitable modelling platform and the effort, cost and complexity of provisioning the compute resources to run them.

10 Practical Demonstrations

In this section we describe our experience of using EC2 to execute a simple agent-based model in the cloud. We illustrate the first two use cases described above – manually configuring a template virtual machine that has been instantiated on EC2, and running a contextualization script to setup and configure a modelling and simulation environment, also running on EC2. We do not include the third use case – using VirtualBox to bundle a machine image before exporting it to EC2 – because this involves a substantial degree of configuration complexity that in large part obviates the benefits of using the cloud and is not a process that most user are likely to engage with. We also compare the process involved with running a batch simulation job using the Jenkins build server (see above) and running the same job using the Model Exploration Service.

For each demonstration we ran the Bankers model, part of [insert name here]'s RepastCity project, used on the [Manchester Social Simulation course].[29] This model is relatively simple but contains a number of more advanced features compared to the introductory tutorials bundled with Repast Simphony. The

10.1 Manual Configuration

Manually configuring a virtual machine with an appropriate simulation environment is similar to the configuration of virtual machines used for other purposes, or the configuration of local non-cloud resources, and does not present any simulation-specific problems. After launching a VM on EC2 and SSHing into it from the command line the installation of the appropriate packages and modelling environment was straightforward. The commands we ran are the same as those contained within the contextualization script contained in appendix [x].

10.2 Configuration through a Contextualization Script

We also used a contextualization script to automatically install and configure a fully functioning modelling and execution environment. This process parallels the manual case, but removes potential configuration errors and reduces the time and effort users must invest before using their virtual machine.

10.3 Exporting Local Machine Images to the Cloud

We chose not to bundle our own a disk image using VirtualBox and exporting it to EC2.

The image used by the [Manchester Social Simulation course] contains Ubuntu Linux and Repast Simphony, the data necessary to run the simulations, and instructions for

[29] https://sites.google.com/site/socialsimulationcourse/home

students on how to use the software. We encountered significant configuration complexity because of the need to ensure interoperability between the virtual machine image file produced by VirtualBox and the virtualization environment supported by EC2.

10.4 The Model Exploration Service (Simulation-as-a-Service)

Gulyas *et al*. (2010) present an on-demand simulation service allowing users to access compute resources over the web – 'Simulation-as-a-Service' – called the Model Exploration Service (MES).[30] Developed by AITIA International Inc., MES is a cloud-based simulation resource that permits users to provision a specified number of CPUs to explore the simulation space of complex agent-based models. Access to MES is through the Model Exploration ModulE tool (MEME),[31] which exposes services compatible with simulation runs executing on common platforms such as NetLogo and Repast J. MEME is part of a larger collection of tools known as the Multi-Agent Simulation Suite (MASS),[32] which supports the development, execution and analysis of agent-based models.

MEME is designed to allow users to execute batch mode simulation runs. At present it has native support for Repast J, which is installed along with the MEME client by default, although it is possible to support NetLogo and pure Java models using MEME's plugin architecture. The data results generated by a particular run are stored in a database and built-in visualisation tools allow users to visualise their data in a number of formats (bar chart, histogram, pie chart, etc.). MEME therefore supports the coordination simulation runs, output management and results analysis.[33] To provide compute power MES uses Amazon EC2 to host virtual machines specifically tailored for agent-based simulations, although exactly how this is done is not specified.
To use MES users need to create an account on the MES web page, and download and install the MEME client.

11 Future Work

Factors to consider:

- Short-term vs. long-term
- Technical vs. academic research directions
- Teaching specific vs. general ABMS cloud use
- Virtual machine disk image repository

---

[30] http://modelexploration.aitia.ai/
[31] http://mass.aitia.ai/intro/meme
[32] http://mass.aitia.ai/
[33] See the MEME manual for full details. The manual is contained in /Users/{user name}/meme/MEME/Documents following a standard installation on OS X or similar location on Linux/Windows.

- E-Learning Simulation Ecosystem
- Specialised cloud simulation services
- Simulation-as-a-Service
- Detailed comparison of cloud service providers
- Tutorials/walkthroughs
- Simulation-specific technical documents on cloud computing

11.1 ABMS E-Learning Ecosystem

There are a number of e-learning proposals in the literature advocate cloud services as the primary delivery channel for course content in different learning contexts. Bhattacharya, Tao, Wu, Qian and Palmer (2011) report a work-in-progress which leverages Google cloud services including Google Apps, Sites, Docs, Accounts, Mail and Custom Search to implement an e-learning suite delivering an interactive, collaborative learning experience for undergraduate introductory computer programming courses. Masud and Huang (2012) also develop an e-learning architecture that uses the cloud.

11.2 Other Recommendations and Future Work

[Any other recommendations here]

12 Conclusion

To model social, cultural and economic processes social scientists employ a battery of methods against a number of practical and theoretical targets. Simulation techniques are used to explain and understand the causal mechanisms that yield the empirical phenomena exhibited by complex multi-agent social systems, predicting when and how social phenomena occur, and to discover novel relationships between social agents and the emergent properties of the structures realised by their inter-relationships (Gilbert and Troitzsch 2005).

There is no doubt that the development of computers in recent years has greatly helped these investigations and that cloud computing has great potential to help social scientists teach social simulation.

13 Appendices

13.1 Sample Contextualization Scripts

This section presents example contextualization scripts, including the scripts we ran in the demonstrations above. These are reusable and easy to adapt. An easily discoverable, deep contextualization script repository would be a significant boon to instructors.

13.1.1 Script 1: Ubuntu, Repast Simphony and User Data

[Script here]

14 References

- See other document in DSR Dropbox folder.