Facilitating the Comparison of Social Simulations using E-science

Bruce Edmonds

Centre for Policy Modelling
Manchester Metropolitan University

bruce@edmonds.name

Abstract. Computational simulations of social phenomena are becoming increasingly common. They embody theories or descriptions concerning such phenomena – varying from abstract analogy down to computational description. However how the simulations algorithm causes the resultant behaviour can itself be only an hypothesis – one that can only be disconfirmed as the result of a computational experiment. It is argued that, regardless of whether social simulations are abstract (KISS) or descriptive in nature (KIDS) that it is vital to extensively replicate and compare simulations in order to determine the nature of the simulation. To rule out a misattribution of cause of a simulation's behaviour the simulations to be compared should be as independent as possible, preferably by different researchers within a different computational environments. Comparing code of different implementations of the same simulation can not guarantee they are essentially the same, this has to be checked via comparing the results. E-science can facilitate such comparison of simulation results by allowing simulations to be mounted so that they can be remotely run and their results queried. One method of doing this is described which allows remote querying via traditional interfaces and agent query languages. This relates to work by others employing the semantic grid.

Why Social Simulations need to be Compared

Computational simulations are being increasingly used as representations of social phenomena. These simulations embody models of their target phenomena – the set-up and initial settings represent the situation at the start and the resulting behaviour as the simulation runs represents the dynamic outcomes. These can be, but are not necessarily numerical in nature (Edmonds and Hales 2004). The models they embody are intended to represent their target phenomena either directly or (more often) indirectly via the intuitions or "mental models" of the researchers who use them (Edmonds 2001). These two approaches can be roughly associated with the KIDS and KISS approaches respectively, as described below. This is but one dimension of a number of issues about which the social simulation community disagree (e.g. the discussion in Conte et al. 2001).

Whatever the kind, a simulation is like a theory in that it aims to capture (descriptively or analogically) some aspect of the causation behind some phenomena. They are formal in nature (being syntactic in nature) but not analytic, in the sense that a general formulation or understanding about their workings can be produced. In other words, how a simulation produces the results claimed for it is itself a hypothesis, accessible only via computational experiment (Edmonds and Hales 2005). It is this aspect of simulations that gives rise to the problems addressed below.

The *KISS* Approach

KISS stands for "Keep It Simple Stupid!", and is supposed to indicate an approach where one starts with a model that is simple as possible, and then only allows for elaboration if the simpler model is shown to be inadequate. As Einstein is reported as saying (though no doubt with tongue firmly in cheek) "*Theories should be as simple as possible, but no simpler*". Partly, this is a matter of control – simpler models are far easier to understand and hence make work. This approach tends to result in very abstract "physics-type" models which are amenable to partial mathematical analysis. These hover (uncomfortably in my view) between pure computer science and computational analogy. As pure computer science they are indisputably legitimate – an exploration of a particular way of organising computation that may be guided by a social interpretation. As such they will be judged in a similar way to mathematics, i.e. their generality, significance, elegance etc.. However, frequently such models are not justified with a view to such criteria but rather by reference to some social phenomena, in other words they do not claim legitimacy via the criteria that might be applied in computer science but rather as a result of their relevance to social phenomena. Thus, it is more common that such models are presented as a computational analogy rather than as computer science. Since they are an analogy they do not relate *directly* to the target phenomena but rather seek to inform our mental "models" concerning these phenomena. This makes it almost impossible to disconfirm in any Popperian sense since one can not disprove an analogy. Instead such an computational analogy is used to join the debates on the phenomena as another narrative, albeit of a computational kind to enrich our understanding (which can have the side effect of endowing it with a spurious status).

Another reason some want simpler models is that they associate such simplicity with generality. This allows their authors to interpret their models in general ways, for example about cooperation in general, rather than about a very particular kind of cooperation represented by a specific model. One suspects that in some cases it is the grandeur of the claims that this allows that drives the modelling approach more than a desire to understand the intricacies of social phenomena. However (unlike perhaps in physics) in the realm of social phenomena there is no reason to suppose that models with a more general applicability will be simpler. In fact, it may well be the reverse – one may get away with simpler models in very specific contexts due to the overwhelming effect of one particular mechanism or process but in general it might be necessary to include a multitude of different mechanisms and processes. If one took a reductionist approach it might only be the incredibly complex biological reality of being human that is truly general.

Another reason to doubt that such abstract *KISS* models have the generality which they are frequently interpreted as having is that often these models are quite brittle, in the sense that if the model is changed in minor ways or slightly extended to take into account more information about the phenomena, the behaviour of the model (in terms of outcomes) changes significantly. If the general interpretation of such a model does not rule out such minor changes, then the fact that the claimed behaviour in such a model disappears under such a change fundamentally challenges the coherence (and hence validity) of such an interpretation. For this reason it is important to test such models using independent replication and "model-to-model" comparison (Edmonds and Hales 2003), where experiments on closely related models explore the conditions under which particular model behaviour occurs.

The *KIDS* Approach

KIDS has been coined in opposition to the *KISS* approach – it stands for "Keep It Descriptive Stupid!" (Edmonds and Moss 2004). This is intended to indicate an approach where one starts with a model which incorporates as much of what is known about the social phenomena as possible. This includes anecdotal accounts, the views of stakeholders and domain experts, as well as more formal data such as transcripts, surveys, and time series. Computational social simulations that follow this approach are often complex, with separate parts of the program representing different observed entities – in other words they tend to be agent-based simulations, since it is often the case that we are concerned with the interactions of observed actors. Such models are much closer to specific descriptions than general theories (in any grand sense). They happen to be specified in the language of computer programs since this is good for representing complicated webs of causation, but they are also amenable to inclusion in public discourses and participatory approaches (e.g. Barreteau et al 2003;. Ramanath and Gilbert 2004).

One approach to this is to specify the agents behaviours (in the model) as the result of a participatory process, incorporating computational analogues of anecdotal evidence as to the behaviour of actors and then repeatedly replaying animations of behavioural outcomes to the participants for criticism and further suggestions. The outcomes can also be checked against more formal data sources, such as time series, by seeing if their "statistical signatures" seem to match. That is that aspects of the data that are deemed important (e.g. unpredictable but pronounced clusters of high activity separated by periods of relative calm) also appear in the corresponding outcomes from the model. Such statistical signatures are not derivable in any *a priori* way using statistical or econometric analysis, but rather spring from an understanding of the social processes that are occurring. This approach is described in more detail in (Moss and Edmonds, in press).

A disadvantage of the *KIDS* approach is that the models are themselves very complex and hence difficult to understand. Thus it is important to experiment with and explore the properties of the model in order that our understanding of it may be increased. In other words one has to model the computational model using information gained from a detailed examination of the runs and controlled computational experiments. Computational experiments involve targeted changes to the code and then an inspection of the outcomes for significant changes. Thus the *KIDS* approach also involves a lot of "model-to-model" comparison, but for different underlying reasons to those in the *KISS* approach.

Thus in either case (KISS or KIDS) there is a need for the comparison of simulations, both in terms of code, but also in terms of the results. If the systems one is comparing them across are at all different, it is likely that the one can not guarantee that different implementations of it are essentially the same without checking the results. This might also require the comparison of simulation data with (potentially) large data sets concerning observations of social phenomena. In all of these tasks e-science can be of help.

The Challenges in Comparing Social Simulations

Checking that simulations will have the same behaviour simply by examining the code is often difficult and, for most encountered classes of simulation, impossible in any general and systematic way. Thus it is almost always the case that whether two pieces of code have the same behaviour (in some specified respect) is a hypothesis that can only be disproved via experiment. Ideally such experiments should be independently replicated by different

researchers using different programming systems and on different computers – this minimises the chance that factors other than the ones described are responsible for the significant results. The trivial case, where the core code is simply copied (rather than reimplemented) and run within another computational environment by other researchers, does not help to check which elements of the code are important in producing the results and which are unimportant implementational details. Better is if the aspects of the code that are considered important are described in abstract terms and a different implementation tested.

In a sense the experimental testing of such simulations is easy, since one has control over almost all aspects of running code, however differences in platform and libraries, as well as the technical knowledge necessary to run simulations in different languages, can present some obstacles to this.

To summarise, the significant results or behaviour of a running simulation might be due to:

- The core algorithm as identified by the original simulator;
- Some other known aspects of the algorithm considered unimportant but included simply to facilitate (or make possible) the simulation;
- Some other aspects of the algorithm that the original simulator did not know existed (e.g. due to bugs or unintended interactions);
- The particular parameter values chosen for the simulation runs;
- The computational environment in which the code is run (e.g. the random number generators from libraries used);

The particular way in which the behaviour of the simulation was analysed or summarised (e.g. a statistical artefact or misleading graph).

The idea is to identify what behaviour is due to the core algorithm rather than any of the other factors. Thus if simulations can be compared varying as many of the other factors as possible, the more confident we can be about claims concerning what outcomes the core algorithm causes. In any case it is desirable to facilitate the comparison of simulations. E-science can play a role in this facilitation.

Alternative Approaches

There are several approaches to facilitating the replication and comparison of social simulations. These include the following:

1. Disseminating descriptions of simulations that are precise enough to allow their independent replication
2. Disseminating results arising from runs of the code with given parameters
3. Disseminating the code itself along with instructions for running it
4. Allowing others to run some reference code remotely and query the results
5. Allowing other to automatically find and interact with some reference code remotely

Clearly these approaches are largely complementary. Currently, if you are lucky, a mixture of techniques 1, 2 and 3 and employed. However this allows for only a limited checking of any replicated simulation and makes it difficult for those without the necessary programme environment to check the simulation. Approach is perhaps the most sophisticated, but currently there is not the middle ware to facilitate this (especially for novice users) – however it is clearly an avenue for further development. The approach we chose was number 4. This is described below.

The Prototype Developed

The basic idea is to use grid and web services to allow the running of simulation code remotely with chosen parameters and then to be able to remotely query the results. This should aid model-to-model comparison and thus to facilitate the replication, criticism and exploration of complex social simulations. Given that large social science data sets might also be made available to remote querying in a similar way, this might also allow automatic comparison of this data with simulation generated data. The key issue is just how automated this can be. In particular will autonomous processes be able to use semantic web technology to automate the use of such a service.

Due to the essentially representational nature of agent-based simulations, there are very deep problems of meaning and reference with respect to the extent to which agents will be able to utilise information from a simulation without a significant level of specific instruction from a human. This basically comes down to the "symbol-grounding problem" (Harnad 1990) which semantic web technology will not solve. However, this does not prevent more limited structures of available data and meta-data being made accessible to facilitate less general approaches, allowing a range of specific tools and agents suitable for computational social simulation researchers to develop and use.

We suggest an approach that uses interfaces the real-time requests on-line (with agents or humans) with off-line simulation. The interface should meditate between the two with a system of XML databases which will store requests for simulations and the corresponding results. Agents/actors can query to see what simulations are available with what parameters and their ranges; query to see what sets of results have been requested and finished; request specific sets of simulation runs and then actively explore the results. Sets of simulation runs can then be cleared from the database when old and hence the total storage capacity can be kept to reasonable limits, whilst still retaining interactive interaction via exploratory queries.

Progress towards this end has been made due to seed funding from the EU 5FP AgentCities program (http://agentcities.net). This involves a webservice in two parts: the first takes either ACL messages or direct method invocation from a Java SWING interface and translates these into XML encoded SOAP messages (as well as, of course) the reverse; the second is an apache SOAP server running on top of Apache Tomcat which takes these XML messages and either runs queries on an XML database (Xindice) or adds data (in the form of a request for a simulation run) to another such database. When there is spare processing capacity the second server takes the next request for a simulation from the database and runs the appropriate simulation, adding the resultant data to the database of results. This database of results can be queried at any time. The use of XML databases should retain the maximum flexibility for future adaptation as well as easy compatibility with developing meta-data formats in the realm of agent-based social simulation (see future directions below).

This structure allows the server with the databases and the mounted simulations (MMU) to be separate from the server that deals with outside requests (Olympus). It means that not all sets of simulation results (which is vast) have to be stored on the server at any one time, but allows for any such set to be dynamically created and queried as required. Some further details about this prototype can be found in the appendix.

Related work

An alternative, and perhaps more sophisticated approach, is being followed in the Fearlus-G project (Edwards et al. 2005; Pignotti et al. 2005a, 2005b; Pohill et al. 2005). There semantic web technology is employed to provide a method of applying meta-data to the various kinds of object used in simulation (a simulation, set of results, hypothesis, etc.) and the simulation and results made accessible via grid technology. The semantic framework provides a structure for an interface to the system and the grid provides a framework to enable the scaling of the approach. The pilot project is based around the FEARLUS simulations. For more about FEARLUS see http://www.macaulay.ac.uk/fearlus/ and the FEARLUS-G project see http://www.csd.abdn.ac.uk/research/fearg/.

The FEARLUS-G approach has a greater potential than the above for automation and discovery because of the semantic web technology and is more easily scaleable due to the grid technology. However the approach described above may be simpler to access and have less overheads. Which is the better approach – making such simulations more accessible to more social scientists – will only become clear in time.

Acknowledgements

Some of this work was done as the result of an AgentCities grant, under the EU 5FP agentcities.net project.

References

Barreteau, O. et al (2003). Our Companion Modelling Approach. *Journal of Artificial Societies and Social Simulation* vol. 6, no. 1 (http://jasss.soc.surrey.ac.uk/6/2/1.html)

Conte, R., Edmonds, B., Moss, S. and Swayer, R. K. (2001). Sociology and Social Theory in Agent Based Social Simulation: A Symposium. *Computational and Mathematical Organization Theory*. Vol. 7 no. 3, pp. 183-205.

Edmonds, B. (2001) The Use of Models - making MABS actually work. In. Moss, S. and Davidsson, P. (eds.), Multi Agent Based Simulation, *Lecture Notes in Artificial Intelligence*, 1979:15-32.

Edmonds, B. and Hales, D. (2003) Replication, Replication and Replication - Some Hard Lessons from Model Alignment. *Journal of Artificial Societies and Social Simulation*, vol. 6 no. 4. (http://jasss.soc.surrey.ac.uk/6/4/11.html)

Edmonds, B. and Hales, D. (2005) Computational Simulation as Theoretical Experiment, *Journal of Mathematical Sociology*, 81(3 or 4) (http://cfpm.org/cpmrep106.html).

Edmonds, B. and Moss, S. (2005) From KISS to KIDS – an 'anti-simplistic' modelling approach. In P. Davidsson et al. (Eds.): *Multi Agent Based Simulation IV*. Springer, *Lecture Notes in Artificial Intelligence*, vol. 3415 pp. 130–144.

Edwards, P., A. Preece, E. Pignotti, G. Polhill and N. Gotts (2005) Lessons Learnt from Deployment of a Social Simulation Tool to the Semantic Grid. 1st International Conference on eSocial Science, 2005, this volume.

Harnad, S. (1990). The symbol grounding problem. *Physica D* vol. 42, pp. 335-346.

Moss, S. and Edmonds, B. (in press) Sociology and Simulation: - Statistical and Qualitative Cross-Validation, *American Journal of Sociology*.

Pignotti, E., Edwards, P., Preece, A., Polhill, G., & Gotts, N. (2005a) Semantic support for computational land-use modelling. Cluster Computing and Grid 2005, Cardiff, UK.

Pignotti, E., P. Edwards, A. Preece, G. Polhill and N. Gotts (2005b). Providing Ontology Support for Social Simulation. 1st International Conference on eSocial Science, 2005, this volume.

Polhill, J. G., Edoardo Pignotti, Nicholas M. Gotts, Pete Edwards, Alun Preece (2005) An ontology for experimentation with a social simulation of land use change. 3rd Annual Conference of the European Social Simulation Association (ESSA), Koblenz, September 2005.

Ramanath, A. M. and Gilbert, N. (2004) The Design of Participatory Agent-Based Social Simulations. *Journal of Artificial Societies and Social Simulation* vol. 7, no. 4 (http://jasss.soc.surrey.ac.uk/7/4/1.html)

Appendix – Outline of Prototype System

Design and Implementation

The implementation of the ISM webservice has been split into two parts: the first takes either ACL messages or direct method invocation from a Java SWING interface and translates these into XML encoded SOAP messages (as well as, of course) the reverse; the second is an apache SOAP server running on top of Apache Tomcat which takes these XML messages and either runs queries on an XML database (Xindice) or adds data (in the form of a request for a simulation run) to another such database. When there is spare processing capacity the second server takes the next request for a simulation from the database and runs the appropriate simulation, adding the resultant data to the database of results. This database of results can be queried at any time. The use of XML databases should retain the maximum flexibility for future adaptation as well as easy compatibility with developing meta-data formats in the realm of agent-based social simulation (see future directions below).

This structure allows the server with the databases and the mounted simulations (MMU) to be separate from the server that deals with outside requests (Olympus). It means that not all sets of simulation results (which is vast) have to be stored on the server at any one time, but allows for any such set to be dynamically created and queried as required. The software used on these two servers is listed in table 1. The structure of the set-up is illustrated in figure 1.

Deployment / *Role*	Basic Run Environment	Main Technologies
User	**1. Windows (browser)**	
Olympus	**1. Linux** **2. Java 1.4** **3. Tomcat v4.1.12**	**1. JSP + JavaBeans** **2. JADE Agents** **3. Apache SOAP**
MMU	**1. Windows/Linux** **2. Java 1.4** **3. Tomcat v4.1.12**	1. Apache SOAP 2. Xindice 3. XMLdbGUI **4.** Simulation

Table 1. The Software used in the ISM service

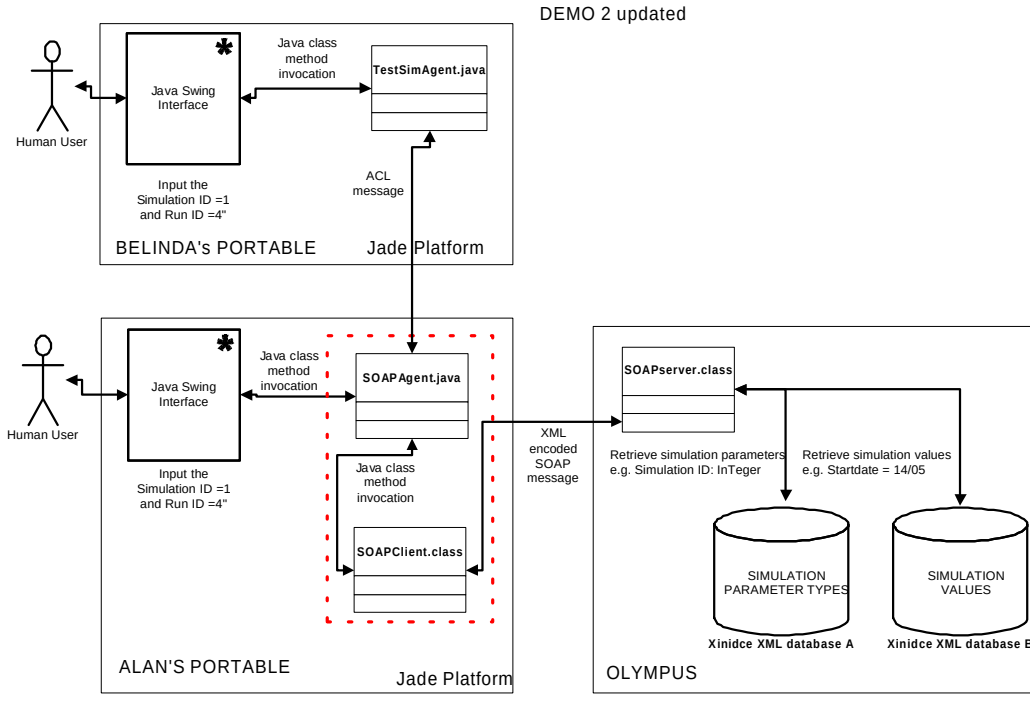

Figure 1. An illustration of the final architecture of the ISM service

The XML databases that mediate between the agent-based simulations and the ACL/web interface service (SOAPserver.class in figure 1), are divided into three:

SimulationModels: this holds the basic information about the simulation models that are mounted – i.e. the parameters that can be set, how they are invoked, the results that are returned, the documentation about them etc.

RunRequests: this holds information about the requests for simulation runs that are made of the server and the status of those runs, so that an agent can discover whether they have been performed as yet, and any error messages.

SimulationResults: this holds the results of the simulation runs, so that they can be queried and extracted remotely by agents and humans.