A new approach to modelling frameworks

J. Gary Polhill & Nicholas M. Gotts
Macaulay Institute, Craigiebuckler, Aberdeen. AB15 8QH.
{g.polhill, n.gotts}@macaulay.ac.uk

Abstract

This paper outlines a new approach to building modelling frameworks (software systems capable of implementing several similar models), based on ontologies. We show how ontologies can act as a bridge between empirical evidence and simulations, by making explicit the assumed links between entities in the real-world and objects in the model.

1. Introduction

One of the features of studying complex social systems is that there is more than one way to describe them. This has the consequence that there is more than one way to model them, which in turn has the consequence that when any one particular model suggests a certain result, the question is always open that another model might have suggested a different result. This problem is related to the concerns raised by Oreskes et al. (1994) in an article arguing that the logic behind validation of models of open systems (in which complex social systems are often situated) is flawed, through committing the fallacy of affirming the consequent (Copi, 1986):

Assume: If a model is valid, then it fits the data.	V -> D
Observe: This model fits the data.	D
Conclude: This model is valid.	∴ V

At the same time, any particular description of a complex social system that takes the form of a computer simulation model is somewhat inaccessible. Human readers of such descriptions must understand the computer programming language in which the software is written and any libraries used. If design documents are not available, it is also necessary to reverse-engineer the program back to the design, to link with the high-level description of the model given in any source text (such as a journal paper). Even if such documents are available, it is still necessary to check that the source code implements the design as expected, as illustrated by Edmonds and Hales's (2003) exploration of Riolo et al.'s (2001) model. This creates an issue for model transparency, which is of particular importance in cases where computer models are part of scientific publications or the democratic process (Polhill et al., in preparation). It is difficult to establish a clear link between the evidence for the model (in the form of high-level, natural language, scenario descriptions) and the model itself.

Modelling frameworks, such as FEARLUS (Polhill et al., 2001; Gotts et al., 2003), are typically used to address the issue of having several alternatives for representation. Modelling frameworks collect together various components that can be combined in different ways to create a particular model, and then explore variants on that model to see how sensitive a result is to the representation scheme used. However, such frameworks, precisely because a number of implementation variants are provided, inevitably involve considerably more source code than would be used for a single model, thereby exacerbating the issue of transparency and link between evidence and model.

Christley et al. (2004) have already argued that ontologies can be used to address three issues caused by the ad-hoc way in which agent-based models are programmed: hidden underlying assumptions, verification of the software implementation, and validation of the conclusions from the model. This paper builds on these arguments to demonstrate how ontologies can be used with modelling frameworks to address some of these issues, although a redesign of the way in which modelling frameworks are built is entailed. After describing ontologies, and how they can be used to establish a link between evidence and models, the paper proceeds to outline a design for a modelling framework to work with them, and discuss how the design can be made more generic.

2. Ontologies

2.1. A brief introduction to ontologies

An ontology is defined by Gruber (1993) as "a formal, explicit specification of a shared conceptualisation". Fensel (2001) elaborates: ontologies are *formal* in that they are machine readable; *explicit* in that all required concepts are described; *shared* in that they represent an agreement among some community that the definitions contained within the ontology match their own understanding; and *conceptualisations* in that an ontology is an abstraction of reality. OWL (McGuinness & van Harmelen, 2004), which stands somewhat anagrammatically for Web Ontology Language, is becoming established as one of the standard languages for writing ontologies, for a number of reasons, not least that it is based on description logics (i.e. formal languages with well-defined semantics), and has a web-compatible syntax (Horrocks et al., 2003). OWL ontologies are already being used by Pignotti et al. (2005) as part of a semantic grid application to manage simulation experiments with FEARLUS, and Christley et al. (2004) present an ontology of agent-based simulation modelling. Since OWL ontologies are founded on description logics, the creator of the ontology effectively asserts a set of axioms in that logic, which collectively constitute the ontology.

An OWL ontology consists of a set of descriptions of *concepts*, a set of *properties* that concepts may or may not have, and a set of *individuals*, which are instances of particular concepts. It is also possible to define *restrictions* on concept membership, which enables automated reasoning software to determine such things as concept satisfiability, and to infer whether or not an individual is an instance of a concept. Concepts are arranged in a hierarchy using the subconcept relation, and are not assumed to be disjoint (as they are in object-oriented programming), though this can be asserted. Concepts can be subconcepts of more than one concept. The subconcept relation means that any individual asserted or inferred to be a member of a subconcept may be inferred to be a member of its superconcepts. Properties can also be organised into a hierarchy using the subproperty relation, in which individuals asserted or inferred to have a particular property may be inferred to have all that property's superproperties. Properties can be thought of as being similar to instance variables in object-oriented classes, though the relationship is somewhat different, as properties have an existence independent of OWL concepts. There are two main kinds of property: object properties allow relationships to be established between individuals belonging to different concepts, and datatype properties allow concepts to have primitive datatypes (e.g. numbers and strings) associated with their member individuals.

Taken together, the concepts and properties can be understood by software engineers as allowing the description of something like a class diagram in UML, and an entity-relationship diagram in relational database design. However, it is the restrictions that give OWL the power to define concepts and allow reasoning software to infer concept membership. There are various kinds of restriction. Both kinds of property may have a cardinality restriction put on them (e.g. a Person may have only one date of birth, or only one biological mother); with OWL allowing specification of minimum, maximum and exact cardinality. For object properties, it is also possible to restrict the concepts involved in the property. Two kinds of restriction are available: one kind says that all values in the property must be from a particular concept (which is also valid if an individual has no value for that property), the other says that some values in the property must be from a particular concept (which is also valid if other values in the property do not belong to that concept). For example in the Person concept, consider the property 'hasBiologicalMother'. All members must have a Person as their biological mother (in OWL this restriction is termed `someValuesFrom`), and all biological mothers of a Person must be a Person (in OWL: `allValuesFrom`).

Standard tools exist to facilitate the development of ontologies, one of the most popular being Protégé, available under a Mozilla Public Licence from protege.stanford.edu. Protégé contains facilities for developing concepts, properties, individuals and restrictions, and plug-in tools for visualisation.

2.2. Using ontologies with modelling frameworks

Consider the situation in which there is a real-world scenario that is being studied. There is some empirical evidence that has been gathered from various sources, and the researchers desire to model some aspect of this scenario; perhaps to explore prognoses for the scenario, to investigate how it could have developed differently, or to compare the potential effectiveness of various proposed policies that might be applied.

In what follows, we demonstrate how four different kinds of ontology could be used together to link evidence from the scenario to what is happening in a model that will be created from a modelling framework in order to allow the exploration of variants on the implementation of certain concepts in the scenario.

Starting with the evidence, the *scenario ontology* describes the concepts applying to a particular scenario or case study. The idea is that this ontology could be developed independently of the other ontologies, the modelling framework or from any particular model one might have in mind. This would be desirable where, for example, the investigative methodology stipulated strictly that the model should be derived from the evidence, rather than the other way round. Where there are no such constraints, the scenario ontology could of course be influenced by the other ontologies and/or design for a model. The scenario ontology may have uses of its own unrelated to modelling in qualitative analysis of case study data, and can be derived from case study texts and other sources of evidence (Polhill & Ziervogel, subm.). The capability of OWL to represent individuals allows the potential for the ontology to be used as a database, perhaps recording questionnaire responses or interview information. A scenario ontology would be created for each investigation to be conducted using the modelling framework, though this does not preclude the possibility that two investigations might share a scenario ontology.

The *domain structure ontology* consists of a description, using an ontology, of the concepts that the modelling framework software implements. The purpose of this ontology is to describe the conceptual context in which the modelling framework sits. It will therefore contain the description of some concepts that do not necessarily appear in the modelling framework, but are related to it in some way. Its creation would quite possibly pre-date that of the scenario ontology, and it is intended to act as a bridge between *any* real-world scenario that the modelling framework might be used with, and the simulated entities in the modelling framework implementation. The domain structure ontology would change with major revisions to the modelling framework, where, for example, such revisions entailed implementations of new concepts.

The *framework ontology* contains concepts from the domain structure ontology as they are simulated by the modelling framework. The separation of the framework and domain structure ontologies allows for the possibility that other modelling frameworks might be used with the same domain structure ontology. The framework ontology contains a description encompassing all the models the framework might be used to implement; in particular, descriptions of implementation variants on specific concepts (such as Case-Based Reasoning (Aamodt & Plaza, 1994), Neural Networks, or other mechanisms for representing the cognition of agents). OWL's web-compatible design allows for ontologies to import each other over the internet. The framework ontology imports the domain structure ontology, adding subconcepts and subproperties to it that correspond to each of the implementation variants of those concepts in the domain structure ontology that it provides implementations for. All concepts in the framework ontology are subconcepts of concepts in the domain structure ontology. The framework ontology would be updated with each non-trivial revision to the framework, e.g. when a new implementation variant is added to the framework.

Finally, the *model ontology* brings the modelling framework and the scenario together, linking the evidence to the model by drawing on both the framework ontology and the scenario ontology. Multiple model ontologies might be required during the investigation of a single case study or scenario, as the focus shifts between aspects of it. To reflect the fact that the model ontology is a specific instantiation of the modelling framework, the model ontology contains a subset of the axioms that the framework ontology adds to the domain structure ontology; this subset collectively constituting a particular choice of implementation variant concepts. The model is presumably intended to reflect some specific aspect of the scenario, so the model ontology needs also to specify how the relevant concepts in the scenario ontology are related to concepts in the domain structure ontology. OWL provides the `equivalentClass`[1] relation for asserting that two concepts are equivalent, and thus the model ontology should import the scenario ontology, and explicitly state which concepts in it are deemed equivalent to concepts in the domain structure ontology. An explicit, transparent link is thereby created from entities in the scenario to their particular implementation in the model: all classes in the model are implementations of subconcepts of concepts in

[1] Somewhat confusingly, OWL refers to concepts as 'classes'. In this paper, the term 'class' is specifically reserved for use with object-oriented programs, and 'concept' for use with ontologies.

the domain ontology that have been declared to be equivalent to concepts in the scenario ontology. The relationships among the various ontologies are shown in figure 1.

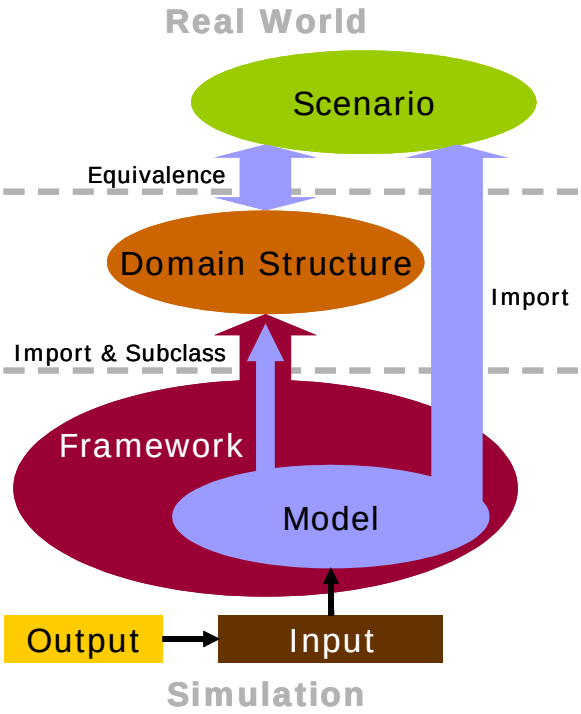

Figure 1. The relationships among the various ontologies. The diagram is expressed as a set of colour-coded elements, each colour corresponding to the contents of a particular ontology. The core of each ontology is rendered using an ellipse, with the relationships it establishes with other ontologies using labelled arrows of the same colour. The input and output system (sub-)states are rendered using rectangles rather than ellipses; this reflects the fact that, though written in OWL, they are not, by themselves, full ontologies, as they contain no concept definitions other than those imported from the model ontology. The black arrow from each system state indicates an import.

Through selecting the particular implementation variants that are to be used in the model, the model ontology makes explicit how concepts are represented in the model, and the framework ontology makes explicit how these representations could have been done differently (though not necessarily exhaustively so). This process therefore also reflects that there could be more than one model for a particular scenario. Further, the fact that the scenario ontology is linked to the domain structure ontology through the OWL `equivalentClass` relation, expresses the fact that this is the ontologists' particular interpretation of the scenario, and signifies that even different modelling frameworks, with different domain structure ontologies, could be used to model it. (The strict independence of the scenario ontology is thus potentially of importance.) The ontologies, taken together, make explicit and transparent how a particular model has been developed, and how it is intended to relate to the area of study. In contrast, the current situation is that the model is treated very much like a black box, to which inputs are given and by which outputs are produced, with relatively little scope for scrutiny and alternative interpretations.

An appropriately programmed modelling framework could then take a model ontology and parameters in an input file containing OWL individuals, and use this to configure a particular run of the framework through initialising the appropriate components. Though written in OWL, since the input file contains only individuals (and no new concept definitions), it cannot be properly thought of as an OWL ontology, but a system (sub-)state. Through establishing a link between the scenario ontology and the domain ontology, it would also be possible to import relevant parts of the

case study data to parameterise certain aspects of the model run. Once the run is complete, the modelling framework could output the set of individuals summarising the state of the system. These could then be used to analyse the fate of individuals from the case study as implemented by the model using standard ontology tools and reasoning services, by comparing the individuals with definitions of concepts in the scenario ontology. Ontology reasoning services could also be used to check the system state summary for consistency against the domain ontology as a means of confirming the validity of the run.

3. Framework design

This section outlines the design of software systems that create modelling frameworks of the kind required to work with ontologies in the manner described above. The approach is based on a distinction between *actors* (objects that do something) and *actions* (what the actors do). Actors are not necessarily people, but could also be inanimate objects that nevertheless, as part of the description of the scenario, are responsible for changes that occur in the system. Other objects may be present not as actors, but as sources of information for the actors to process in the course of their actions.

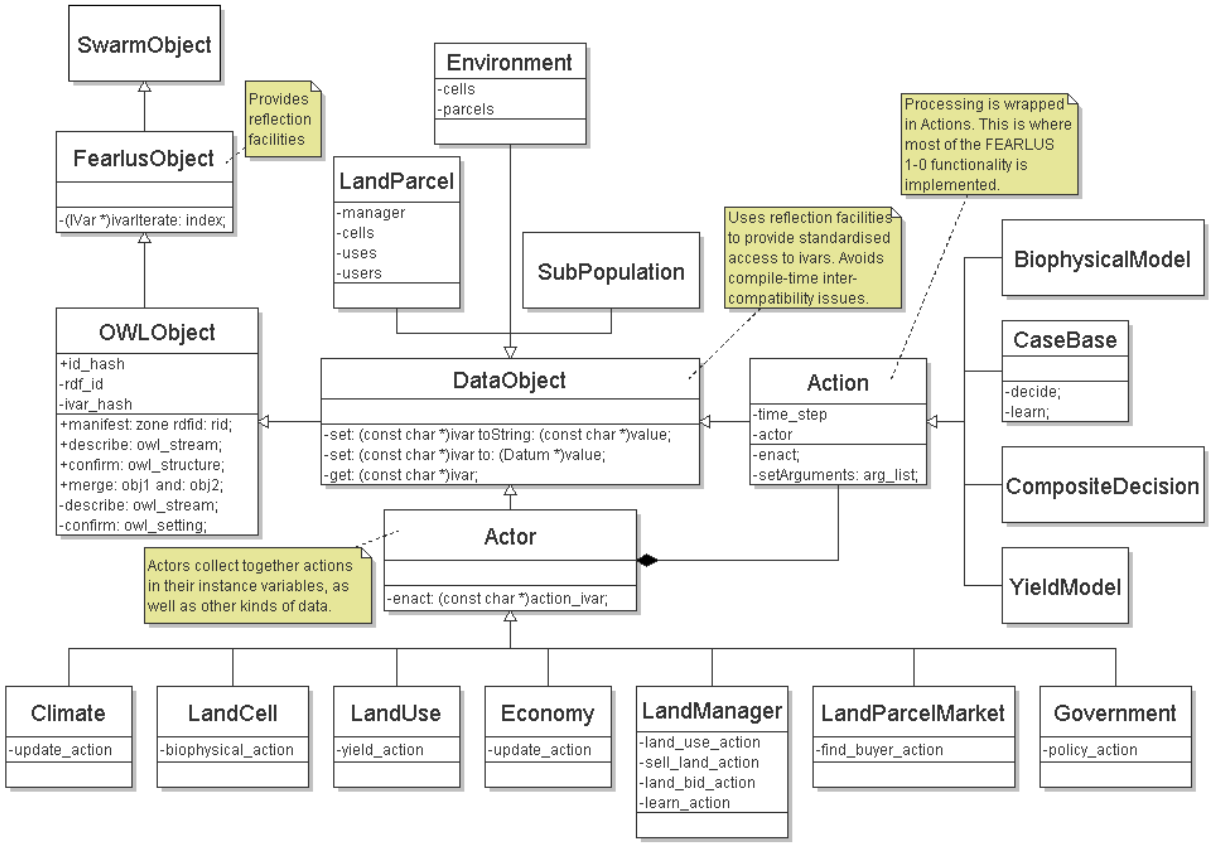

Figure 2: A UML class diagram for FEARLUS 1-0. Note that a more Obj-C-like syntax has been used for the instance variables and methods than in standard UML.

A UML class diagram indicating the proposed design for FEARLUS 1-0 is outlined in figure 2. The core inheritance hierarchy from DataObject through to SwarmObject establishes the creation of classes that can read and write their instances from and to OWL files (with the possible exception of the Environment class, which may need to read information from a GIS), and provide generic accessor methods to their instance variables. Classes like the Environment, LandParcel, and SubPopulation act as information repositories, storing data of a certain kind that will

be used by the Actions. The model ontology will specify which of these classes are entailed in a particular simulation, and a class not involved will need to disable the creation of members to enforce the stipulations of the ontology. Furthermore, specific instance variables will correspond to properties (datatype or object) in the ontology, and if these are not explicitly specified, the class will need to disable access to them for the same reason that members of classes not in the ontology should be prevented from being created. Thus the accessor methods provided by the DataObject class, though they involve an overhead on computation, play an essential part in enforcing the model ontology, thus ensuring that the ontology is a transparent reflection of the model structure.

The Actor class is the superclass of all classes of object that cause a change in the state of the system. Each Actor is effectively a special kind of DataObject, with some instance variables containing Actions of one kind or another. These Actions are enabled through another standard interface method. Actors do not, however, contain methods to perform the Actions. The Actions are performed by the Action object with which they are associated. Actions may also need to store data that can be considered a part of the state of the system (such as a case base for a case based reasoner), and so are also a subclass of DataObject. Actions obtain the data they need to do their actions from the Actors with which they are associated, using the accessor methods provided by the DataObject class. These methods are also used to change the state of any objects that are affected by the Action.

4. Discussion

The fact that all access to the Actors and DataObjects is through standardised accessor methods suggests a stronger sense of encapsulation than is the case in standard object-oriented programming, indicating that an Agent-Oriented design approach might be more suitable (Wooldridge et al., 2000). Software agents could be used to represent individuals, which would communicate with each other using an appropriate language such as KQML (Knowledge Query and Manipulation Language; Finin et al., 1994)—an approach used in PALM (Matthews, in press). For small and medium-scale simulations not requiring distributed processing due to resource limitations on a single machine, the case for a design based on software agents is rather less strong, and whilst the conceptual link to AO design exists, there is little to be gained from the investment required to develop the code in this way.

The *specificity* of a modelling framework can be seen as the set of models it can implement. A partial ordering of modelling frameworks can be established through the *more-specific-than* relation, in which framework F1 is more-specific-than framework F2 if the set of models that F1 can implement is a proper subset of the set of models that F2 can implement. A related concept is that of the number of real-world scenarios that the framework can be appropriately applied to. The more models a framework can implement, the greater the potential for its use with any particular scenario. We may thus also consider the *more-applicable-than* relation, in which F1 is more-applicable-than F2 if the set of scenario ontologies that F2 can be used to build a model from is a proper subset of the set of scenario ontologies that F1 can. The design of FEARLUS 1-0 discussed above involves hard-coding certain classes and instance variables that are intended to correspond to concepts and properties in the domain ontology. This constrains somewhat the set of scenario ontologies to which FEARLUS 1-0 can be related. Where concepts in the scenario ontology, though similar to concepts in the domain ontology, contain too many differences in properties, the use of the `equivalentClass` assertion in OWL to establish a link between them will be open to criticism. The modularity of the design, separating information sources, actors and actions, enables the system to be extended relatively easily. Extra Actions can be added to the library of Actions without affecting the other Actions, or necessarily affecting Actors or DataObjects unless extra kinds of information are required by the new Actions. Since the Actors and DataObjects contain only instance variables, new classes of Actor or information source can be added quickly and easily to the system, as can new instance variables to existing classes.

Although the ontologies enable a transparent description to be made of the structure of the model—the objects and the relationships between them—the processes by which the state of the system is changed, although explicitly labelled, have their mechanisms hidden from view in the library of Actions. This situation can be ameliorated somewhat by including more configurable Actions, such as a case based reasoner with configurable constituents of cases. However, there are a couple of issues with developing a system that also includes specification of process details, typically involving an algorithm. Firstly, in the most general case, the ontology would then simply become a substitute for the model; some of the advantages of simplifying the view of the model would be lost as the descriptions start to include more technical algorithmic terms such as iterations, sequences, and selections. Secondly,

it might be desirable for the algorithms themselves to be based on established scientific theory, in which case it is of more importance to record the provenance of the theory, and any debate about it (something an ontology could perfectly well make explicit). Where there are several alternative theories for a particular process, a good modelling framework would make provision for them, enabling the exploration of the effects of using the alternatives on the results.

In this paper, we have shown how, with a new approach to designing modelling frameworks, ontologies can act as a bridge between empirical evidence and social/socio-ecological simulations. Specifically, the ontologies enable explicit statements to be made about the links between concepts in the modelling framework, and concepts in the real world.

5. Acknowledgements

This work has been co-funded by the Scottish Executive Environment and Rural Affairs Department, and by the EU Framework 6 NEST Pathfinder Initiative on Tackling Complexity in Science.

6. References

Aamodt, A., & Plaza, E. (1994), Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications* **7** (1), 39-59.

Copi, I. M. (1986) *Introduction to Logic. Seventh Edition.* New York: Macmillan.

Christley, S., Xiang, X., & Madey, G. (2004) Ontology for agent-based modeling and simulation. In Macal, C. M., Sallach, D., & North, M. J. (eds.) *Proceedings of the Agent 2004 Conference on Social Dynamics: Interaction, Reflexivity and Emergence, co-sponsored by Argonne National Laboratory and The University of Chicago, October 7-9*. http://www.agent2005.anl.gov/Agent2004.pdf

Edmonds, B., & Hales, D. Replication, replication and replication: Some hard lessons from model alignment. *Journal of Artificial Societies and Social Simulation* **6** (4). http://jasss.soc.surrey.ac.uk/6/4/11.html.

Fensel, D. (2001) *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*. Berlin: Springer.

Gotts, N. M., Polhill, J. G., & Law, A. N. R. (2003) Aspiration levels in a land use simulation. *Cybernetics and Systems* **34** (8), 663-683.

Finin, T., McKay, D., Fritzson, R., & McEntire, R. (1994) KQML: An information and knowledge exchange protocol. In Fuchi, K., & Yokoi, T., (eds.) *Knowledge Building and Knowledge Sharing*. Ohmsha and IOS Press.

Gruber, T. R. (1993) A translation approach to portable ontology specifications. *Knowledge Acquisition* **5**, 199-220.

Horrocks, I., Patel-Schneider, P. F., & van Harmelen, F. (2003) From SHIQ and RDF to OWL: The making of a Web Ontology Language. *Web Semantics: Science, Services and Agents on the World Wide Web* **1** (1), 7-26.

Oreskes, N., Shrader-Frechette, K., & Belitz, K. (1994) Verification, validation, and confirmation of numerical models in the earth sciences. *Science* **263**, 641-646.

Matthews, R. B. (in press) The People and Landscape Model (PALM): An agent-based spatial model of livelihood generation and resource flows in rural households and their environment. *Ecological Modelling*.

McGuinness, D. L., & van Harmelen, F. (2004) OWL Web Ontology Language Overview. *W3C Recommendation 10 February 2004*. Latest version: http://www.w3.org/TR/owl-features/

Pignotti, E., Edwards, P., Preece, A., Polhill, G., & Gotts, N. (2005) Semantic support for computational land-use modelling. *2005 IEEE International Symposium on Cluster Computing and the Grid. Cardiff, Wales, May 9-12, 2005 Volume 2*. Piscataway, NJ: IEEE Press. pp. 840-247.

Polhill, J. G., Gotts, N. M., & Law, A. N. R. (2001) Imitative and nonimitative strategies in a land use simulation. *Cybernetics and Systems* **32** (1-2), 285-307.

Polhill, J. G. & Ziervogel, G. (subm.) Using ontologies with case studies: an end-user perspective on OWL. *Submitted to the Second International Conference on e-Social Science, 28-30 June 2006, Manchester, UK.*

Polhill, J. G., and others (in preparation) Software licensing issues in social simulation. *Draft document available from the author*.

Riolo, R. L., Cohen, M. D., & Axelrod, R. (2001) Evolution of cooperation without reciprocity. *Nature* **411**, 441-443.

Wooldridge, M., Jennings, N. R., & Kinny, D. (2000) The Gaia methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems* **3** (3), 285-312.