International Environmental Modelling and Software Society (iEMSs)
2012 International Congress on Environmental Modelling and Software
Managing Resources of a Limited Planet, Sixth Biennial Meeting, Leipzig, Germany
R. Seppelt, A.A. Voinov, S. Lange, D. Bankamp (Eds.)
http://www.iemss.org/society/index.php/iemss-2012-proceedings

An ontology-based design for modelling case studies of everyday pro-environmental behaviour in the workplace

Gary Polhill [1], Nick Gotts [1], Noelia Sánchez-Maroño [2], Edoardo Pignotti [3], Óscar Fontenla-Romero [2], Miguel Rodríguez-García [2], Amparo Alonso-Betanzos [2], Pete Edwards [3] and Tony Craig [1]
[1] The James Hutton Institute, Aberdeen, UK.
[2] Facultade de Informática, Universidade da Coruña, A Coruña, Spain.
[3] Department of Computing Science, University of Aberdeen, Aberdeen.
[1] gary.polhill@hutton.ac.uk

Abstract: This paper outlines the design of a model for simulating six case studies of everyday pro-environmental behaviour in the workplace. Rather than creating six separate models, we want to exploit similarities among the case studies and maximise code re-use – a problem common in the world of integrated modelling. Noting concerns about modular integrated modelling that have been raised by numbers of authors, we start from a standpoint of viewing integrated modelling as a problem of semantic integration. A system for integrated modelling using OWL ontologies as the medium of representation of the structure and state of the model at any one time is presented and an early prototype implementation of the case study model evaluated.

***Keywords**:* semantic integration; model integration; ontologies.

1 INTRODUCTION

The LOCAW (Low Carbon at Work: Modelling Agents and Organisations to achieve Transition to a Low Carbon Europe – http://www.locaw-fp7.com/) project aims to identify how carbon consumption practices in the workplace and the home can be transformed, and to enhance our understanding of how these two important areas of our lives can be made to work together to achieve a transition to a sustainable society. It involves case studies of six organisations in the public and private sector, including services and heavy industry.

Modelling multiple case studies requires a modular approach with reusable algorithms implementing model dynamics. At the same time, each case study needs to be an internally consistent model in its own right. The research area of building models through integrating modular reusable components has been on-going for several years now, particularly in the area of human-environmental system modelling. Established approaches to software integration, such as component-based approaches [Bian and Hu 2007] and software agents [Wooldridge and Ciancarini 2001], have so far placed greater emphasis on encapsulation (data hiding) than standard object-oriented design, and many integrated modelling frameworks, including the Open Model Interface Environment (OpenMI) [Moore and Tindall 2005; Gregersen et al. 2005] adopt this approach in one form or another. There are good reasons to do so, such as facilitating legacy software reuse, and preventing misuse of submodels' internal variables.

However, greater encapsulation means less integration. In the terminology of Antle et al. [2001], submodels are loosely coupled through connecting the output variables of one submodel to the inputs of others. Whilst semantic annotation of input and output variables goes some way to addressing concerns over their inappropriate linkage [e.g. Rizzoli et al. 2008], there are wider issues with what

might reasonably be characterised a 'black box' integration approach. Reflections on model coupling in various areas of research have concluded that loose coupling can lead to problems with ontological consistency in the coupled whole [e.g. Frysinger et al. 2002; Leavesley et al. 2002].

These issues were covered recently by Voinov [2010], who used a series of images as metaphors for concerns he had over model integration issues. Borrowing one of these images, we can illustrate conceptually one of the issues with semantically-labelled black box integration (figure 1). Semantic integration is a live issue in the database community, where it occurs in the integration of distributed heterogeneous databases – for example, when companies merge and their personnel databases need to be integrated. Bellatreche et al. [2006] list common problems that come under the heading of semantic heterogeneity: *naming conflicts* (where the same name is used for different entities, or different names for the same entity), *scaling conflicts* (where concepts are represented at different spatial or temporal scales), *confounding conflicts* (where concepts appear to have the same meaning, but don't), and *representation concepts* (where concepts are represented in different ways).

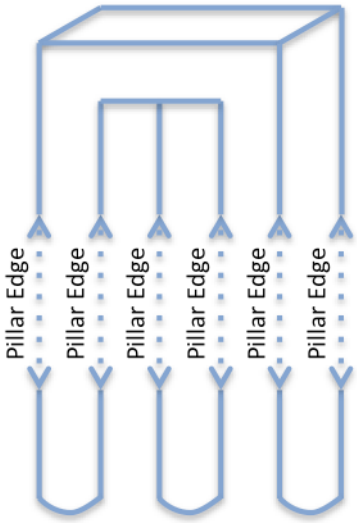

Figure 1. Redrawing a famous optical illusion to illustrate issues with semantically labelled loose coupling.

Software integration is no less a problem of semantic integration than is database integration, and therefore, so is model integration [Polhill and Gotts 2011]. Indeed, we have argued [ibid.] that model integration is a more difficult semantic problem than database integration, because of what could be termed *algorithmic conflicts*. These arise because concepts in models are not only represented by their descriptive properties (classes, attributes, attribute value constraints), but also by algorithms representing their dynamics: the processes by which the attributes change value over time. Algorithmic conflicts occur when two submodels need to represent the same subprocess in order to compute their output variables, but that subprocess is implemented in different ways. The presence of the subprocess in each submodel could be completely hidden if all that is visible is the input and output variables of the submodels. Although black-box coupling can avoid algorithmic conflicts by ensuring that each submodel operates in a distinct, non-overlapping domain of the whole simulated system, adherence to this constraint cannot be verified or enforced without prior in-depth knowledge of the computations performed by each submodel. Without such knowledge, the possibility of unknown ontological conflicts (such as economic growth and recession in the same area at the same time) cannot be ruled out. Hence, in addressing issues of semantic heterogeneity in model integration, it is arguable that to ensure ontological consistency we should explore approaches using less, not more, encapsulation.

2 MODELLING IN THE LOCAW PROJECT: WERC-M

The LOCAW project uses agent-based simulation models as a synthesis tool and as part of a backcasting exercise, in which the case study organisations are asked to think about where they want to be in the context of a low-carbon future [Sánchez-Maroño et al. 2012]. The models are then used to provide simulated narratives that lead, or do not lead, to that desired outcome. The case study organisations selected are: a university, a local government, a water company, the renewable energy arm of a power generation company, a manufacturer of lorry cabs, and an oil company.

The project is focused on everyday practices in the workplace pertaining to the use of energy and materials, management and generation of waste, and transport. An initial study of all six case studied suggested that a core model should be focused around the relevant choices the agents make on an everyday basis, as different case studies entail different degrees of autonomy for agents. For example, lecturers and students in the university enjoy considerably more autonomy than do factory workers in the lorry cab manufacturer.

Figure 2 shows the initial ontology for WERC-M (Worker-Environment Reinforcement Choice Model). In it, the agents are `Persons`, who find themselves in a series of `Contexts`, each `Context` providing them with a set of `Options` among which they have to choose. Each `Option` has an `Impact` on the `Environment`, and gives `Feedback` to the `Person` doing it. Other `Persons` with whom the `Person` choosing the option has `interpersonalRelationships` may observe the choice the `Person` made, and also give `Feedback` on it. The `Feedback` will be used to adjust the likelihood that the `Person` repeats the `Option` in that `Context`.

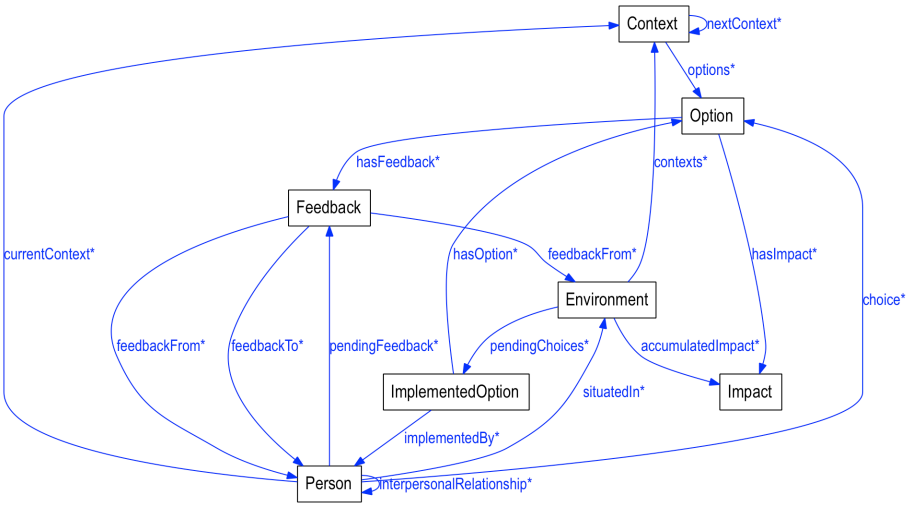

Figure 2. OntoViz graph of the WERC-M ontology.

Each case study will involve a specialisation of this ontology, with its own unique combination of instances of `Option`, `Context`, `Feedback` and `Impact`, subclasses of `Person`, and subproperties of `interpersonalRelationship`. However, the case studies will also have similarities as well as differences, and to this extent, there is scope for overlap in implemented processes from one case study to the next. The issue is similar to the problem of integrated modelling discussed earlier, in that there will be various submodels, some specific to a case study, some shared with other case studies, some common to all case studies, but in each individual case study, whatever combination of submodels are used, we want to be sure that the case study model is ontologically consistent.

In the rest of this paper, we use an early prototype to show how our modelling environment (OBIAMA) could be used to implement the case study models. Reimplementation of models is widely recognised as best practice in agent-based modelling [e.g. Galan and Izquierdo 2005], and a parallel implementation of WERC-M is being built in Repast Simphony [North et al. 2007].

3 MODELLING WITH OBIAMA

OBIAMA is a tool currently in development at the James Hutton Institute. It is designed to be an integrated modelling environment that uses OWL (Web

Ontology Language [Cuenca Grau et al. 2008]) ontologies to represent the state and structure of the model at all times. Sublanguages of OWL are based on decidable description logics, allowing the use of an OWL reasoner, such as Pellet [Sirin et al. 2007] to check the consistency of the model. This is important, as representing the state of the model using an OWL ontology breaks encapsulation. OWL cannot represent dynamics in a model, so changes to the state from one time step to the next are still implemented in Java code. Clearly, the data this code operates on, since they are stored in a separate OWL ontology, are no longer stored with the code. Hence, rather than relying on encapsulation to ensure ontological consistency (which works only if each submodel operates on distinct, non-overlapping, areas of the model), OBIAMA relies on the reasoner.

Building a model using OBIAMA entails the following steps:

- Constructing a model structure ontology. The model structure ontology contains mostly T-box (the 'T' is for 'Terminology') axioms. In description logics, T-box axioms are axioms that describe entity types, properties entities have, and relationships among them. The equivalent in Java would be classes, fields and associations.
- One or other of:
    - Constructing an initial model state ontology. A model state ontology consists entirely of A-box (the 'A' is for 'Assertion') axioms. In description logics, A-box axioms are axioms that describe individuals. The equivalent in Java would be instances.
    - Constructing an initial schedule that builds the initial state. OBIAMA provides a schedule ontology. Building a schedule entails defining instances of concepts in that ontology, which describe a sequence of actions to run that make changes to the state as required.
- Constructing a main schedule that will embody the simulation proper. This, like the initial schedule (if used), is implemented as a series of A-box axioms using terminology in OBIAMA's schedule ontology.

A typical initial schedule consists of a sequence of actions; a typical main schedule consists of a repeated sequence of actions. Each action changes the A-box assertions in the model state ontology, and is implemented by a Java class, which will be written and compiled before the OBIAMA model is built. Hence it will not necessarily have access to the terminology used in the model structure at compile time. To deal with this, each action implementation has its own 'micro-ontology': terminology describing the classes, properties and relationships that it expects to appear in the model structure ontology. OWL provides `EquivalentClasses` and `EquivalentProperties` axioms that can be used where the model structure ontology does not use the same terminology as the micro-ontologies of the actions in the schedule(s). Thus we may add the following to the list of activities involved in building a model:

- Providing implementations for actions, where these are not provided by an OBIAMA built-in action implementation.
- Adding `EquivalentClasses` and `EquivalentProperties` axioms to the model structure ontology where this does not use the same vocabulary as the micro-ontologies of the implementations of the actions in the schedule(s).

A consequence of the loss of encapsulation is that it is possible that no action implementation is ever operating on all the properties of an individual. This creates an issue whenever a new instance of a class is created. Whilst this could be dealt with by building an action implementation to create an instance of each class, this is inconvenient. Instead, classes can be annotated in the model structure ontology with a creator schedule to execute each time an instance of the class is created. A creator schedule consists of a series of actions the implementations of which

provide initial values for all the properties of the created instance. Creator schedules need to be 'inheritable', in the sense that OBIAMA runs any creator schedules of all superclasses of a class of which an instance is being created, as well as the creator schedule of the class itself. Hence, if instances of any class are to be created by actions in the schedule (something that is particularly likely in the initialisation schedule), building an OBIAMA model also involves the following:

- Building creator schedules for any classes of which instances are to be created during the execution of the initialisation or main schedules, and adding them as annotations to the model structure ontology.

In an agent-based model, agents may ask each other for information whilst making decisions. This information is computed by one agent in response to a request by another, and could be specific to properties of both. A trivial example is: "How much older are you than me?" Here again, the lack of encapsulation means that there is nowhere to locate a method that would perform such a computation and return the answer to the requesting agent. It would be extremely unwieldy to implement an action that stored all the differences in ages among all the agents in a reified relationship, when all that is required is an intermediate variable. OBIAMA provides queries for this purpose. Queries do not change the state of the model, but provide derived information from it to other actions. Since this derived information is not stored in the state ontology, the transparency of the model is undermined. Thus, if an instance is to respond to a query during the execution of an action implementation requiring it, one of the classes to which it belongs must have an annotation to that effect. There is therefore the following additional activity when building a model:

- Annotating classes with queries that their instances respond to if the execution of action implementations so requires.

4 PROTOTYPE IMPLEMENTATION OF WERC-M IN OBIAMA

A prototype implementation of WERC-M has been built using OBIAMA, in which agents make a random selection of `Option`s in each `Context`. Choosing these `Option`s results in `Impact` that is accumulated by an instance of `Environment`, and in `Feedback` to the agent. This model provides the following specialisations of the WERC-M ontology shown in section 2:

- `Context` has the instances '`choosingCommutingTravel`', '`stoppingForLunch`' and '`stoppingForDay`', representing the three contexts in which agents will be choosing options.
- Option has the instances '`walking`', '`cycling`', '`bus`', '`driving`' as options for the '`choosingCommutingTravel`' `Context`, and '`on`' and '`standBy`' as options for the '`stoppingForLunch`' and '`stoppingForDay`' `Context`s. The latter are intended to represent agents trying to save energy in the workplace for equipment they are not using while away from work.
- `Impact` has the data property '`pollution`', and instances for each `Option`.
- `Feedback` has the data property '`inconvenience`', and instances for each `Option`.
- `Environment` has the instance '`environment`'.

This specialisation is just for this initial prototype implementation. In LOCAW, we expect each case study to define its own specialisations that are appropriate to the everyday actions and consequences it is investigating. For example, in the case study of the Spanish university, the `Context`s are `goingToWork`, `goingHome`, `attendingAClass`, `teaching`, `takingABreak`, `havingLunch`, `studying` and `researching` [Sánchez-Maroño et al. 2012]. The specialisations for each case study will be developed in collaboration with field researchers on the project.

The schedule ontology has an initialisation schedule and a main schedule, each with their own URI. The actions in each schedule are implemented by a Java class that adds and/or removes A-box axioms from the model state ontology. The initialisation schedule:

- Creates some agents. The `Person` class in the model structure ontology is annotated with two creator schedules: one that assigns the agents an initial context (‘`choosingCommutingTravel`’) and another that sets the environment of the agent.
- Loads values for the ‘`inconvenience`’ of each `Feedback` from a CSV file.
- Loads values for the ‘`pollution`’ of each `Impact` from a CSV file.

The main schedule repeats the following sequence:

- Agents make a random selection among the `Options` for the `Context` they are in.
- Agents add the `Option` they have chosen to the `pendingOptions` of the `environment`.
- The `environment` accumulates `Impact` from and sends `Feedback` for each of the `pendingOptions`.
- Agents update their `Context`.

The result of running OBIAMA with this simple prototype model is a series of model state ontologies, one for each step in the model. In what follows, we consider the implementation of two of the actions described briefly above.

The action to update the `Context` of each agent is one of the simplest in the model, and is coded in the schedule ontology as illustrated in Figure 3:

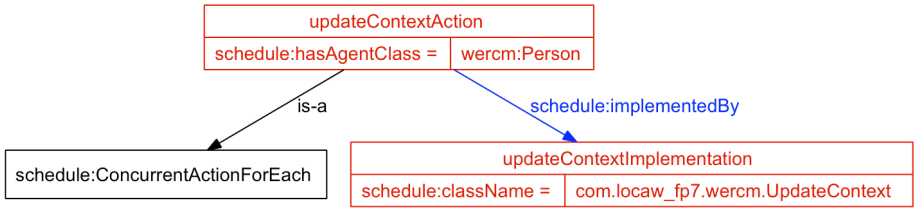

Figure 3. Schedule ontology assertions for the update context action.

This tells the scheduler to run the Java class `UpdateContext`’s `step()` method concurrently for each member of the class `Person`. (For now, ‘concurrently’ means in a random sequential order, but no two actions may ‘interfere’ i.e. write to the same values.). The `UpdateContext` class implements one other method besides `step()`: `initialise()` is called when the schedule is loaded to create the action’s ‘micro-ontology’. The micro-ontology is not explicitly written in OWL, but provides a statement of what the action expects in the model structure ontology; in this case using the vocabulary of WERC-M (see Figure 1):

- an object property `currentContext` with domain `Person` and range `Context`,
- and an object property `nextContext` with domain `Context` and range `Context`.

The `step()` method takes the URI of the agent running the action as argument. It gets the `currentContext` of the agent, and sets it to the `nextContext` of that `Context`.

`UpdateContext` as described can be used for any case study version of WERC-M where the sequence of `Contexts` in the model is linear. However, in some case

studies, it is possible that this constraint will not apply. For example, there could be multiple `nextContexts` that depend on properties of the `Person`, or there could be a need to synchronise certain `Contexts` across agents regardless of what prior `Contexts` they have been in. New `UpdateContext` actions would be required to implement this – these can be coded separately and specified as implementations of the `updateContextAction` in the schedule, without affecting the code implementing other actions.

The action to create the agents in the initialisation schedule is a built-in action implemented by the `CreateNAgentsAction` class, and demonstrates the use of creators. Figure 4 shows the extract of the schedule ontology containing assertions about the `createWorkersAction`. This is an `IndividualAction`, meaning it is performed by a named agent, in this case, the `exogenousAgent` defined in the schedule ontology.

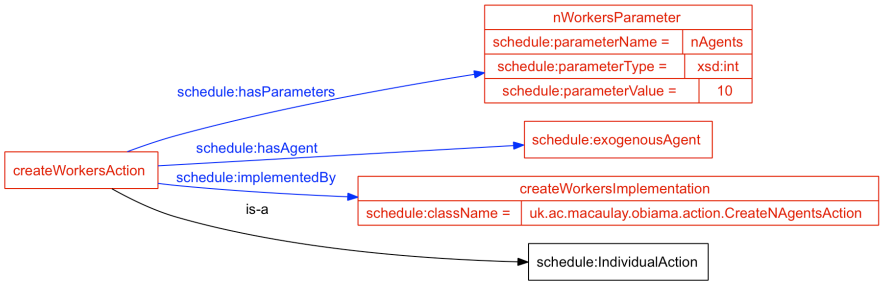

Figure 4. Schedule ontology assertions for the createWorkersAction.

The micro-ontology of the `CreateNAgentsAction` consists only of the class `CreatedThing`, which is expected to appear in the model structure ontology. To make this happen, `CreatedThing` is declared equivalent to `Person` in the model structure ontology. In the case studies, we might expect there to be different subclasses of `Person` – e.g. `Student`, `Lecturer` and `Administrator` in the Spanish university case study – each of which is to have instances created separately. If these are all declared equivalent to `CreatedThing`, then it will be inferred that a `Student` is equivalent to a `Lecturer`, which would be wrong. To get round this problem, OBIAMA provides a facility for actions to rename the base URI of the micro-ontologies of their implementations.

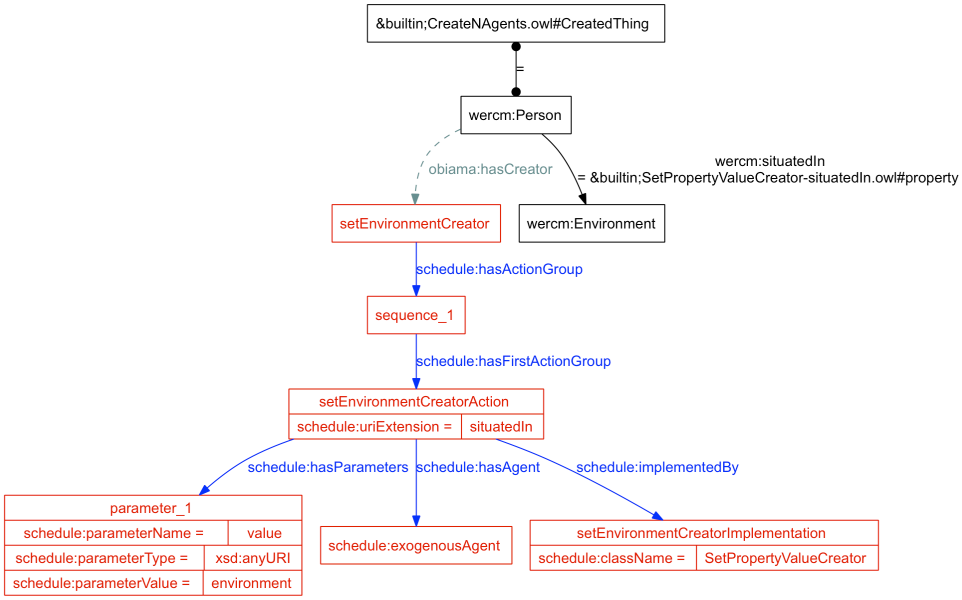

Figure 5. Part of the simple prototype model structure ontology showing the creator schedule annotation of the `wercm:Person` class.

The `CreateNAgentsAction` implementation simply generates URIs for the specified number of individuals, and creates class assertion axioms for each of them. This is all very well, but the agents need initial values for some of their properties: one to set the value for `currentContext`, and another to set the value for `situatedIn`. To do this, the `Person` class is annotated with the `hasCreator` property (Figure 5), defined in the OBIAMA ontology. The range of the `hasCreator` annotation property is a schedule to run whenever an instance of the domain class is created. In the case of `Person`, the creators use the `SetPropertyValueCreator` class to set `currentContext` to `choosingCommutingTravel`, and `situatedIn` to `environment`. Figure 5 illustrates the latter. Micro-ontology renaming is also required: the micro-ontology of the `SetPropertyValueCreator` consists simply of the property of which the value is to be set in the created entity, which must be declared equivalent to the corresponding property in the model structure ontology. To avoid `situatedIn` and `currentContext` being equivalent, the micro-ontology is renamed using the `schedule:uriExtension` property of the action (see Figure 5).

In some of the case studies, it may be preferable to set the initial `Context` of each Person from a file. This can be done easily, and without interfering with any other code implementing the model, by providing a different creator schedule that does not use `SetPropertyValueCreator` as its implementation.

5 DISCUSSION

The use of OWL ontologies in agent-based modelling and related areas is growing. Christley et al. [2004] describe an OWL ontology of agent-based modelling approaches, and argue that such ontologies can assist with exposing hidden underlying assumptions in models, among other things. Guizzardi and Wagner [2010] have developed an ontology of discrete event simulations. Parker et al. [2008a] developed the MR POTATOHEAD framework to capture the components that might be expected to appear in an agent-based model of land use and cover change, and have used it to compare agent-based models of land use change in frontier regions [Parker et al. 2008b]. It has also been implemented in OWL [Parker et al. 2008c]. Müller [2007] uses ontologies in the initial stages of model development to describe a conceptual model with stakeholders, which are then used to develop the UML diagrams from which the object-oriented simulation model is eventually coded. Other relevant work includes Villa's [2001] model integration architecture, which uses XML to describe the modular integrated components, and Rizzoli et al. [2008] who link models using OWL. Bian and Hu [2007] emphasise the use of a standard ontology in a discipline to facilitate model interoperability. The usefulness of ontologies as a medium for representing model state and structure is not confined to integration consistency checking. Transparency [Polhill and Gotts 2009], and the potential for linking the model to text [Gotts and Polhill 2009] are also advantages of this approach.

Whilst OBIAMA enables modular components to be integrated to build models that are nevertheless consistency checked, it is, however, not without its disadvantages. Not least of these is that the radically different approach to programming means that it is not trivial to use legacy software as submodels. It may be possible to write a wrapper action around the legacy software, but ideally this would ensure that variables used in intermediate computations would be exposed – something that is not possible if access to the source code is not available, or the software licence otherwise prohibits it.

The semantics of exchanging data among integrated submodels is not confined to datatypes. There is also the issue of units. Some integrated modelling environments put conversion wrappers around the input/output variables of components, but in the context of OBIAMA, it cannot be assumed that two actions both referring to temperature, for example, will be using the same units. Whilst this issue could be handled using SWRL rules between data properties using different

units, many reasoners do not implement SWRL rules, and many that do only provide a partial implementation.

Another significant disadvantage is that using the reasoner each time step to generate inferred axioms means that running the model is slow. Running the simple version of WERC-M with only 10 agents for 100 time steps took around an hour and a half using the Pellet reasoner. The availability of approximate reasoning tools such as TrOWL [Thomas et al., 2010] provides the possibility that there may be ways to work around the reasoning issues. Other short-cuts may be possible. For example, reasoning could be restricted to axioms associated with A-box assertions in the state ontology. Further, provided a model does not rely on inferences for the purpose of updating variables, consistency checking could be done off-line, after the model has finished running, and in parallel, using one node for each time step.

The discussion above suggests a number of criteria on which software aimed at facilitating model integration can be evaluated:

- Facilitation of software re-use. This has two related elements: the re-use of legacy software (i.e. not built with the model integration software in mind), and the re-use of modular submodels (which have been so constructed).
- Ontological consistency checking. A model that is not ontologically consistent is invalid; it is imperative that model integration software is able to provide a consistency checking service. A subset of issues under this heading are possible conflicts in units of data properties.
- Feasibility. The time and memory cost of integration should not be prohibitive in comparison with building and running a completely new model tailored to the task in hand.

6 CONCLUSION AND RECOMMENDATIONS

Seen as a problem of semantic heterogeneity, the role of semantics in integrated modelling efforts is emphasised. We suggest that emphasising encapsulation when building models from modular components detracts from integration, as algorithmic conflicts can occur across the black-box submodels. Our prototype OBIAMA ontology-based modelling system is able to exploit reasoning capabilities of formal ontology languages to check consistency as a model built from coupled submodels progresses. However, further work is needed to improve the efficiency of the system if it is to be applied effectively to the case studies in LOCAW.

ACKNOWLEDGMENTS

This work was funded by the European Commission through Framework Programme 7, grant agreement number 265155 (LOCAW – Low Carbon at Work: Modelling Agents and Organisations to Achieve Transition to a Low Carbon Europe), the Economic and Social Research Council, grant reference RES-149-25-1075 (PolicyGrid II), and the Scottish Government Rural Affairs and the Environment Portfolio Strategic Research Theme 4 (Economic Adaptation).

REFERENCES

Antle, J.M., Capalbo, S.M., Elliott, E.T., Hunt, H.W., Mooney, S. and Paustian, K.H., Research needs for understanding and predicting the behavior of managed ecosystems: Lessons from the study of agroecosystems, *Ecosystems 4*, 723-735, 2001.

Bellatreche, L., Xuan Dung, N., Pierra, G. and Hondjack, D., Contribution of ontology-based data modeling to automatic integration of electronic catalogues within engineering databases. *Computers in Industry 57*, 711-724, 2006.

Bian, L. and Hu, S., Identifying components for interoperable process models using concept lattice and semantic reference system. *International Journal of Geographical Information Science 21*(9), 1009-1032, 2007.

Christley, S., Xiang, X. and Madey, G., Ontology for agent-based modeling and simulation. Proceedings of the Agent 2004 Conference on Social Dynamics: Interaction, Reflexivity and Emergence, ANL/DIS-05-6, pp. 115-125, 2004.

Cuenca Grau B., Horrocks I., Motik B., Parsia B., Patel-Schneider P., Sattler U., OWL 2: the next step for OWL. *Journal of Web Semantics 6*, 309–322, 2008.

Frysinger, S., Integrative environmental modelling. In Clarke, K.C., Parks, B.O., Crate, M.P. (eds.) *Geographic Information Systems and Environmental Modeling*. Prentice Hall Upper Saddle River, NJ, pp. 221-222, 2002.

Galan, J.M. and Izquierdo, L.R., Appearances can be deceiving: lessons learned re-implementing Axelrod's 'Evolutionary Approach to Norms', *Journal of Artificial Societies and Social Simulation 8*(3), 2.

Gotts, N.M. and Polhill, J.G., Narrative scenarios, mediating formalisms, and the agent-based simulation of land use change. *Lecture Notes in Computer Science 5466*, 99-116, 2009.

Gregersen, J.B., Gijsbers, P.J.A., Westen, S.J.P. and Blind, M., OpenMI The essential concepts and their implications for legacy software. *Advances in Geoscience 4*, 37-44, 2005.

Guizzardi, G. and Wagner, G., Towards an ontological foundation of discrete event simulation. Proceedings of the 2010 Winter Simulation Conference 5-8 December 2010, Baltimore, MD, pp. 652-664, 2010.

Leavesley, G.H., Markstrom, S.L., Restrepo, P.J. and Viger, R.J., Modular approach to addressing model design, scale, and parameter estimation issues in distributed hydrological modelling. *Hydrological Processes, 16*, 173-187, 2002.

Moore, R.V. and Tindall, I., An overview of the Open Modelling Interface and Environment (the OpenMI). *Environmental Science and Policy 8*, 279-286, 2005.

Müller, J.P., Mimosa: Using ontologies for modeling and simulation. Complex07, 8th Asia-Pacific Complex Systems Conference, 2-5 July 2007, Gold Coast, Australia, 2007.

North, M.J., Howe, T.R., Collier, N.T. and Vos, J.R., A declarative model assembly infrastructure for verification and validation, Advancing Social Simulation: The First World Congress, Springer, Heidelberg, Germany, 2007.

Parker, D.C., Brown, D.G., Polhill, J.G., Deadman, P.J. and Manson, S.M., Illustrating a new 'conceptual design pattern' for agent-based models and land use via five case studies: The MR POTATOHEAD framework. In: Lopez Paredes, A. and Hernandez Iglesias, C. (eds.) *Agent-Based Modelling in Natural Resource Management*. pp. 23-51, 2008a.

Parker, D.C., Entwisle, B., Moran, E., Rindfuss, R., Van Wey, L., Manson, S., Ahn, L., Deadman, P., Evans, T., Linderman, M., Mussavi, S.M. and Malanson, G. Case studies, cross-site comparisons, and the challenge of generalization: Comparing agent-based models of land-use change in frontier regions. *Journal of Land Use Science 3*, 41-72, 2008b.

Parker, D.C., Polhill, J.G. and Mussavi Rizi, S.M., An OWL (Web Ontology Language) representation of the MR POTATOHEAD agent-based land-use change meta-model. American Association of Geographers Annual Meeting, Boston, MA, April 15-19, 2008c.

Polhill, G. and Gotts, N., Semantic model integration: an application for OWL. ESSA 2010, Seventh Conference of the European Social Simulation Association, Montpellier, France, September 19-23, 2011.

Polhill, J.G. and Gotts, N.M., Ontologies for transparent integrated human-natural system modelling. *Landscape Ecology 24*, 1255-1267, 2009.

Rizzoli, A.E., Donatelli, M., Athanasiadis, I.N., Villa, F. and Huber D., Semantic links in integrated modelling frameworks, *Mathematics and Computers in Simulation 78*, 412-423, 2008.

Sánchez-Maroño, N., Alonso-Betanzos, A., Fontenla-Romero, O., Bolón-Canedo, V., Gotts, N.M., Polhill, J.G., Craig, T. and García-Mira, R., An agent-based prototype for enhancing sustainability behavior at an academic environment, 10th Conference on Practical Applications of Agents and Multi-Agent Systems Salamanca, 28th - 30th March, 2012.

Sirin, E., Parsia, B., Cuenca Grau, B., Kalyanpur, A., and Katz, Y., Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics 5*(2), 51–53, 2007

Thomas, E., Pan, J.Z. and Ren, Y. TrOWL: Tractable OWL 2 reasoning infrastructure. *Lecture Notes in Computer Science 6089*, 431-435, 2010.

Villa, F. Integrating modelling architecture: a declarative framework for multi-paradigm, multi-scale ecological modelling. *Ecological Modelling 137*, 23-42, 2001.

Voinov, A., Integronsters and the special role of data. Proceedings of the International Congress on Environmental Modelling and Software, Ottawa, ON, Canada, 5-8 July 2010.

Wooldridge, M. and Ciancarini, P., Agent-oriented software engineering: the state of the art, *Lecture Notes in Computer Science 1957*, 55–82, 2001.