Combining Object-Oriented Programming and Relational Databases for Multi-Scale Spatially-Integrated Agent-Based Models

J. Gary Polhill [a], **Nicholas M. Gotts, and Alistair N. R. Law**
Macaulay Institute, Craigiebuckler, Aberdeen. AB15 8QH
[a] *g.polhill@macaulay.ac.uk*

Abstract: Object-oriented (OO) programming has limitations when used to implement abstract multi-scale, spatially-integrated, agent-based models, that could potentially be addressed using relational databases (RDB). Although this would involve rethinking the approach to designing such models, the combined OO-RDB approach has a number of appealing advantages for multi-scale simulations, such as allowing the user rather than the programmer to specify the scale at which various land-use processes take place. It also provides a basis for a more realistic representation of the relationship between agents and their environment.

Keywords: Spatially Explicit Agent-Based Models; Relational Database Modelling; Object-Oriented Design

1 INTRODUCTION

This paper concerns the development of abstract, multi-scale models of land use change using spatially-integrated agents. The purpose of modelling in this way is generally to understand the processes operating in a particular topic rather than to accurately represent a particular scenario for the purposes of prediction or forecasting. Consequently, abstract models tend not to make a great deal of use of real-world data.

When abstract models include elements of scale, it is desirable for the user to vary the scales at which particular processes operate, in recognition of the awareness among some authors [e.g. Nelson, 2001] that scale can be an observer-defined phenomenon. This enables exploration of the effect of simulating processes at different scales on emergent model phenomena. It also provides the flexibility to explore a wider range of scenarios. However, varying the scale at run-time is tricky, because the objects concerned with scale-related processes cannot know at compile-time the scale at which such processes will function.

Few land use models attempt multi-scale approaches. The CLUE modelling framework [Veldkamp & Fresco, 1996] is one exception. Their scale hierarchy is fixed, however, and totally ordered — though the resolutions to which each scale applies can be varied to fit the scenario and availability of data [Verburg et al., 1999].

The FEARLUS (‘Framework for the Evaluation and Assessment of Regional Land Use Scenarios’) project explores the use of spatially-integrated agents (representing land managers) to improve understanding of land use change. The approach used has been to start with simple, abstract designs, and build in complexity and elements of realism subsequently [Polhill et al., 2001]. The current version of the model (0-5) consists of a rectangular grid of land parcels and a set of heterogeneous land manager agents with various (simple) algorithms for the selection of a land use. The agents accumulate wealth from their land parcels according to how well the selected land use matches the (spatially variant, but temporally constant) biophysical characteristics of the parcels, the climate, and the economy. The climate and economy change from year to year in the model, but are spatially invariant.

The paper proposes an architecture (as yet unimplemented) that enables agents to be designed without specific reference to the spatial scales at which their behaviour is taking place. This uses a relational database [Codd, 1970] to store all the publicly available knowledge, with the various scales of spatial representation integrated using relational tables. The paper also questions some assumptions about the role of encapsulation within agent-based modelling.

2 THE APPEAL OF OBJECT-ORIENTED PROGRAMMING FOR AGENT-BASED MODELS

The appeal of object-oriented (OO) design is based on the four core elements embodied in most OO programming languages: *Abstraction* (deciding what will be included in the simulation), *Encapsulation* (collecting the properties and behaviours of each class of object in a single data structure, hiding the internal workings from the rest of the program), *Inheritance* (arranging classes into a hierarchy) and *Polymorphism* (allowing the same behaviour to be implemented differently in different classes) [Hunt, 1998].

The OO paradigm has been recognised as being highly suitable for building agent-based models [Gilbert & Troitzsch, 1999], although it has been observed that pure OO design methodologies are not adequate to represent and describe the specialised set of concepts and inter-relationships that are needed when designing agent-based models [Wooldridge et al., 2000; Kinny et al., 1996]. Nevertheless, agent-based models are usually implemented in an OO programming language, and agents can be seen as a subset of objects [Luck & d'Inverno, 2001].

However, one feature of the OO paradigm is the distributed nature of the data about properties of objects. Whilst some properties of agents relate to their internal states, others, though they are still specific to the agent, are such that other agents requiring information about them should not necessarily need to ask the agent for them. Examples include spatial location and physical appearance. These external properties are particularly relevant in spatially-integrated models because the space provides the external context in which they may be observed.

When multiple spatial scales are involved, the OO paradigm has limitations that make implementation difficult. The development of FEARLUS is towards a more flexible approach to the representation of the various scales at which processes influence land use change. For example, allowing the user to specify that climate could vary spatially, or that land use should be chosen at the farm, rather than the parcel level. This means providing a structure to enable the agents to process information and make changes to their world, ideally without explicitly coding all the different options for obtaining the information that the user might configure. Therefore, there is a need to distinguish between public (or external) and private (or internal) properties of objects, and to provide an architecture for the design of agents in the absence of explicit knowledge of the spatial scales at which their behaviour operates.

3 THE PROBLEM OF SCALE

Scale is used to mean a number of different, but often interlinked, things. In a geographical context scale typically refers to the level of detail of observation or representation (resolution) of a particular spatial region. The resolution of the data affects the interpretation and recognition of spatial features and patterns. These form the basis of another way of looking at scale, in which the land is divided into parcels according to a classification scheme applied to the observed properties. The parcels so formed will have a smaller average area when there are more classes in the classification scheme.

A hierarchy of scales can enable consideration of parcels of land defined in terms of a particular property, which can be subdivided into smaller parcels that are associated with another property. For example, farms (the parcels of land associated with a particular owner) always consist of one or more fields (the parcels of land associated with a particular crop). Alternatively, a particular property (such as land cover) may be described by a class hierarchy, and a scale hierarchy could also be formed to mirror this. For example, a forest land cover class can be sub-classified into evergreen, deciduous and mixed forests, which are distinguishable at different levels of resolution.

The scale hierarchy will be partially ordered when the properties of the land do not form a single chain of sub-divisions of parcels, an example of which may be land ownership and soil type (Figure 1). This adds an extra dimension of complexity that multi-scale simulations typically either ignore or work around using totally-ordered scale hierarchies that have neighbouring parcels with the same property value.

In FEARLUS, a modelling framework is being built that allows the user to select which properties and processes (pertaining to land use change) they wish to simulate. They should also be able to determine how these properties and processes will relate to one another in terms of scale, *i.e.* to configure the scale hierarchy at run time.

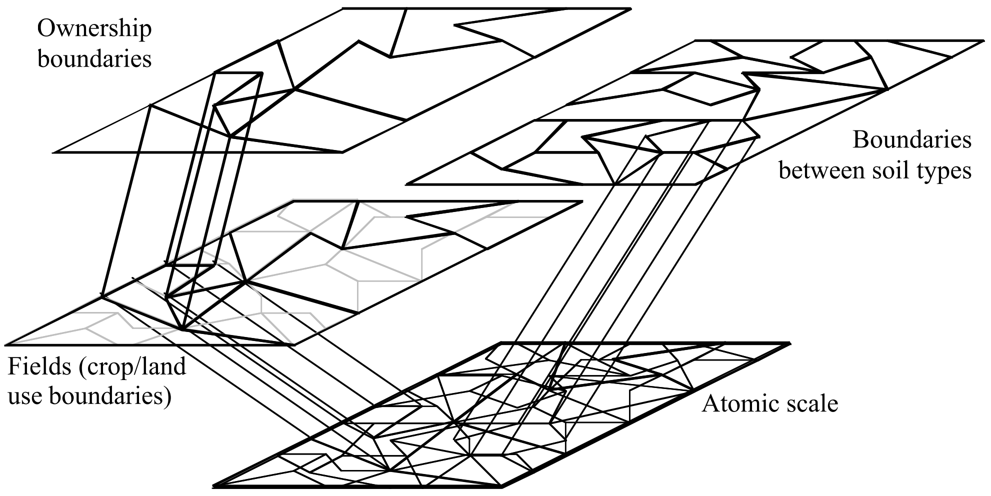

Ownership boundaries
Boundaries between soil types
Fields (crop/land use boundaries)
Atomic scale

Figure 1. An example of a hierarchy of scales associated with different properties of the land

From the OO design point of view, none of these problems present obstructions to representation. In an abstract model, one could consider something like a "ScaledParcel" class, which would have an anonymously typed "landProperty" instance variable to store the spatial data associated with that scale, a ScaledParcel-typed instance variable containing the parent scale parcel, and a list of child parcels, if applicable. Both totally and partially ordered scale hierarchies could be represented in this way (partially-ordered scale hierarchies requiring a list of parent parcels and a list of lists of child parcels).

If the model is abstract, however, implementation in the OO paradigm presents difficulties when the model is required to be sufficiently flexible to allow phenomena to be configured to take place at various different scales. An example is of farmers deciding what crop to sow. In one model, the user might want a farmer to decide a single crop for the whole farm, with biophysical properties affecting yield varying at the field scale. In another, farmers might be given the capability to choose crop sowings at the sub-field scale (*i.e.* possibly subdividing the field into two separate crop sowings). To provide this kind of functionality within a single modelling framework means that the classes used to simulate farmers cannot have 'hard-coded' access to data about land at a particular scale. This means providing run-time configurable scale-related data transactions within each class.

The problem for OO in implementing scale-related data transactions that can be configured at run-time is that there is no way to relate the various scales together except through the scale hierarchy. Since the scale hierarchy is itself created at run-time, there is no easy way to specify how data should be accessed at compile time, except through exhaustively specifying all the possible configurations of the scale hierarchy and coding for each option individually — a prospect that is combinatorially worse for more complex modelling frameworks that identify a greater number of feature or entity scales.

In non-abstract models, the scales will be defined by the data sources, so the scale hierarchy can be hard-coded. It is possible, however, that in the future, those sources of data will be at a different resolution, which could affect the scale hierarchy. The flexibility to adjust the scale hierarchy may therefore apply to non-abstract models as well.

4 THE APPEAL OF RELATIONAL MODELLING

The OO approach lacks a constructive way of linking the scales in the hierarchy that circumvents the need to exhaustively specify all the user-configurable options for simulating scale. Relational modelling offers a way around these limitations. Once spatial data is stored using relational tables, the possibility arises of storing other data from the simulation in this way, including data about agents.

In the relational model, one table is used to represent each scale of property. A relational table is then created for each adjacent pair of scales in the scale hierarchy (Figure 2). The relational join operator can then be used to link any two scales by searching through the partial ordering of scales. This allows the creation of abstract hierarchies of scale at run-time with easy retrieval of relevant information. Once the user has configured the scale hierarchy, the operations needed to join each

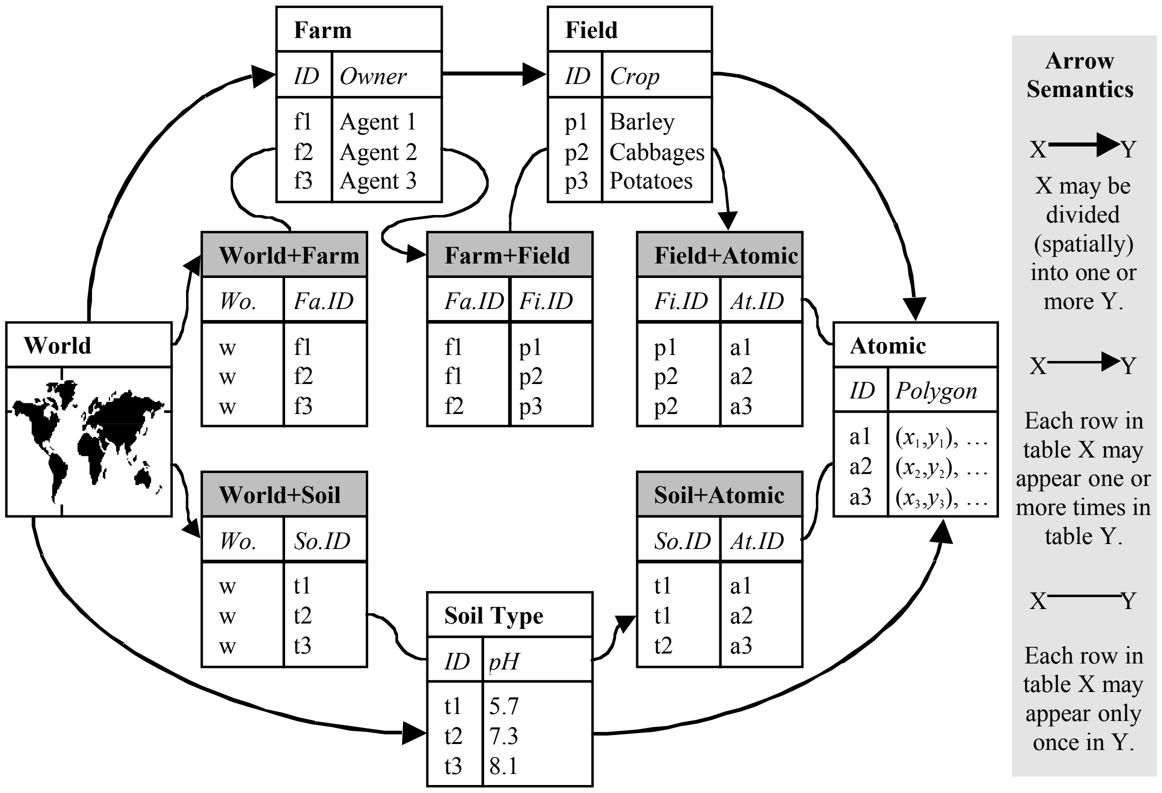


Figure 2. The scale hierarchy (thick arrows) is converted into a series of relational tables (indicated in grey) that link adjacent scales in the hierarchy together (thin arrows). The atomic scale consists of a set of polygons covering the space, with each polygon containing the same value for all scales in the hierarchy.

pair of scales together can be automatically (and exhaustively) generated. Since it will be known at compile-time which operations agents will need to do, at run-time, agents can be configured to use the appropriate automatically generated operation for each time they get information about or update the properties of the land.

Consider, for example, the case in Figure 3. An agent requires access to climate information for the land it owns. The user might configure a number of different scale hierarchies, however, which prevent a consistent operation from being coded in the Agent class. (Two such hierarchies, A and B, are shown in the diagram.) The code in the Agent class consists of a call to the database to get the climate information. The database is configured at run time to construct appropriate join operations for the scale hierarchy the user has configured, returning to the agent a list of the parcels of land containing the information at the highest possible scale.

A relational model enables the representation not only of the various scales of the land and its properties, but also of the relationships between them. This allows abstract representation of land properties to be configured at various scales from model to model within a single modelling framework, without exhaustive explicit coding of all the different scale scenarios in classes that should not really be concerned with them.

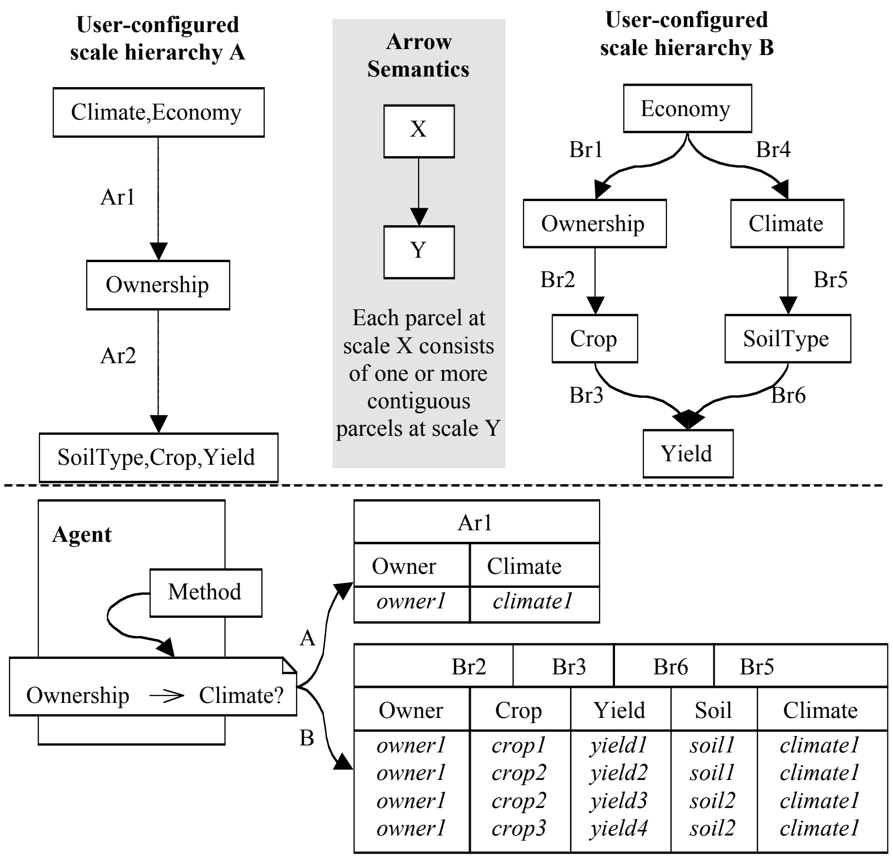
Storing data about agents as well as the space in a relational database could provide a means of representing the external properties of agents discussed earlier. It would also have the benefit of facilitating the construction of reports from the model. However, it would mean a break from strict encapsulation, since the external properties of agents would not be stored within the Agent class.

5 DISCUSSION

This proposal for the design of an abstract, spatially-integrated, agent-based model allows user-configurable scale hierarchies to be enabled using a relational model. It also suggests a departure from core OO design philosophy in the design of agents, through breaking the concept of strict encapsulation. Some of the practical benefits of encapsulation (such as being able to recode one class without affecting others) will be lost as a consequence.

A more serious problem for this proposal concerns the performance implications of replacing a local look-up in the object data structure with a database query. If a query has to be constructed each time the agent needs some information, the model will take considerably longer to run. One possible solution is to construct the queries at initialisation time, and have the queries persist for the duration of the model run. This would have performance implications with each change to the data, but access times would be reduced. Constructing queries at initialisation time would also have implications for adaptive agents, who might change their queries during the course of the run. The obvious solution is to have the queries constructed when first used, and persist until the object creating them is destroyed. In general, however, although there may be techniques for reducing the performance costs, inevitably models built in this way will take longer to run.

Agents are not objects, but subjects interacting with a single objective world of which they are a part. Thus, there is a sense in which even the private internal states of an agent form a part of that world. At some point, however, the question of abstraction arises, and the direct physical effects of agents adjusting their internal knowledge of their environment (if indeed there are any) will not be simulated. Furthermore, in the class hierarchy of Agents, there may be sufficient difference in the way behaviour is derived (say, decision trees in one class and neural nets in another) that it makes more sense for these internal states to be encapsulated within the object, and not stored in a table. The key issue is to realise that strict encapsulation lies at one extreme of a dimension of design, and that this extreme position may not necessarily be best suited to all agent-based models. Indeed, many object-oriented programming languages provide syntax to allow objects direct access to instance variables, which are exploited by authors such as Axtell [2000] in agent design.



Ar1	
Owner	Climate
owner1	*climate1*

Br2	Br3	Br6	Br5	
Owner	Crop	Yield	Soil	Climate
owner1	*crop1*	*yield1*	*soil1*	*climate1*
owner1	*crop2*	*yield2*	*soil1*	*climate1*
owner1	*crop2*	*yield3*	*soil2*	*climate1*
owner1	*crop3*	*yield4*	*soil2*	*climate1*

Figure 3. Two user-configured scale hierarchies (A and B), and an Agent who needs to find out the weather on its property in one of its methods. The relevant join operations are set up at run-time, when the model is initialised, and the Agent connects to these operations to get the information required. The arrow labels are the names of relational tables linking adjacent scales together. In hierarchy A, the query about climate and ownership involves traversing Ar1 to look up the climate for the given owner. In hierarchy B, a more complex query is required, linking two separate chains in the partially ordered hierarchy — the crop, yield and soil type data are not used — these are just the spatial units of land that have to be traversed to get from land ownership data to climate data according to the way the land is divided up in hierarchy B.

These arguments are in apparent contradiction to the views of leading authors in the field of agent-oriented design such as Wooldridge, Jennings and Kinny [2000], who state that "the agent paradigm is based on a significantly stronger notion of encapsulation than the object paradigm" [p. 307]. However, their focus on the strictly social aspects of agent interactions (through concepts such as roles, responsibilities and protocols) does not consider the interactions between agents and their environment. For such 'disembodied' agents, basing design principles on strict encapsulation is, perhaps, the best way forward. The suggestion made here, however, is that once agents are located in a space (not something that Wooldridge et al. are especially concerned with), this principle is turned on its head. Space provides a context for indirect agent communication, through (in the real world) such things as visual, aural and olfactory signs that, although they are properties of the agent, can be observed through the space without directly querying the agent, and so are also, in a sense, properties of the space. Locating agents within a space therefore reduces the sense of their strict individuality. In more general terms, agents are intricately bound with their environment, in that the environment and the agent cannot properly be regarded as being wholly separate.

It is the contention of this paper that of the four elements embodied in most object-oriented programming languages, strict encapsulation need not be considered 'core' to the development of spatially-integrated agent-based models. Using relational modelling as a means to address issues in the representation of spatial scale, which has also been demonstrated in this paper, provides a basis for simulating both the individual agent view and the world view, and the somewhat blurred relationship between them.

This technique was developed to deal with a perceived problem with implementing user-configured spatial scale hierarchies. The scope for applying this technique to temporal and organisational scale hierarchies has not been considered, but should not necessarily be ruled out, as there is nothing inherently spatial in the methodology. So long as a scale hierarchy can be defined with particular kinds of property associated with each scale, and there is an atomic scale to join chains in a partially-ordered hierarchy, there is no immediately obvious impediment to applying relational modelling to temporal and organisational scales as well as spatial. Further work, however, is needed to confirm this.

6 ACKNOWLEDGEMENT

This work is funded by the Scottish Executive Environment and Rural Affairs Department.

7 REFERENCES

Axtell, R. Why agents? On the varied motivations for agent computing in the social sciences. *Center on Social and Economic Dynamics Working Paper No. 17*, November 2000.

Codd, E. F. A relational model of data for large shared data banks. *Communications of the ACM*, *13*(6), 377-387, 1970.

Gilbert, N. and Troitzsch, K. G. *Simulation for the Social Scientist*, Open University Press, 273 pp. UK, 1999.

Hunt, J. *Java and Object Orientation: An Introduction*, Springer-Verlag, 473 pp., London, 1998.

Kinny, D., Georgeff, M., and Rao, A. A methodology and modelling technique for systems of BDI agents. *MAAMAW 96*, *LNAI*, *1038*, 56-71, 1996.

Luck, M. and d'Inverno, M. A conceptual framework for agent definition and development. *The Computer Journal*, *44*(1), 1-20, 2001.

Nelson, A. Analysing data across geographic scales in Honduras: detecting levels of organisation within systems. *Agriculture, Ecosystems and Environment*, *85*(1-3), 107-131, 2001.

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

Veldkamp, A. and Fresco, L. O. CLUE: a conceptual model to study the Conversion of Land Use and its Effects. *Ecological Modelling*, *85*(2-3), 253-270, 1996.

Verburg, P. H., de Koning, G. H. J., Kok, K., Veldkamp, A., and Bouma, J. A spatial explicit allocation procedure for modelling the pattern of land use change based upon actual land use. *Ecological Modelling*, *116*(1), 45-61, 1999.

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