

Adaptive Intelligent Modelling for the Social Sciences: Towards a Software Architecture

Catriona Kennedy and Georgios Theodoropoulos
School of Computer Science, University of Birmingham, UK
{cmk,gkt}@cs.bham.ac.uk

Abstract

This is a progress report on the Computer Science input to the AIMSS project (Adaptive Intelligent Model-Building for the Social Sciences). The work involves the building of an exploratory prototype to apply the paradigm of “Dynamic Data-Driven Application simulation” (DDDAS) to a housing case study in the Public Policy domain. The prototype is composed of four components: an agent-based simulation, a database, some data mining tools and a software agent (known as a “Discovery Assistant” or DA). The DA agent helps to interpret the simulation predictions and configures the data mining tools to determine whether the predictions are supported by the available data. If there is a discrepancy, the simulation should be adapted so that it becomes increasingly consistent with the observations in the data.

1 Introduction

This is a work-in-progress report on the AIMSS project ¹, which is about the application of “Dynamic data-driven application simulation” (DDDAS) to social systems. DDDAS has already been introduced in an earlier report [10] but will be summarised briefly here. In the physical sciences, DDDAS is a method where data from a physical system is absorbed into a simulation of the system [4]. There is a feedback system involving the following processes:

- (G1) The states predicted by the simulation can play a role in selecting the data to be absorbed. In some cases the physical system itself may be modified as a result of simulation predictions. This is also known as symbiotic simulation [15].
- (G2) The predictions of the simulation are continually adjusted by absorption of new data. The underlying model on which the simulation is based may be revised as a result of data assimilation. This is the “data-driven” component.

These are labelled G1 and G2 because they are general components of a DDDAS architecture. Different stages in the implementation of DDDAS are schematically shown in Figures 1, 2 and 3 respectively. Figure 1 shows a simple system where data is continually absorbed into a simulation and the simulation’s states are adjusted as necessary (corresponding to G2 only). In Figure 2, the simulation states help to determine which sensors are activated and how they are directed (implementing both G1 and G2). Figure 3 shows symbiotic simulation.

1.1 AIMSS: A Social Science Assistant

For the social sciences, we are developing a modified DDDAS architecture called AIMSS (Adaptive Intelligent Model-building for the Social Sciences). A schematic diagram is in Figure 4. This is a modification of the “assistant agent” scenario in [10]. U is the user-defined model underlying the simulation, D is the meta-data description of what is in the simulation and the available data sources.

¹<http://www.cs.bham.ac.uk/research/projects/aimss/>

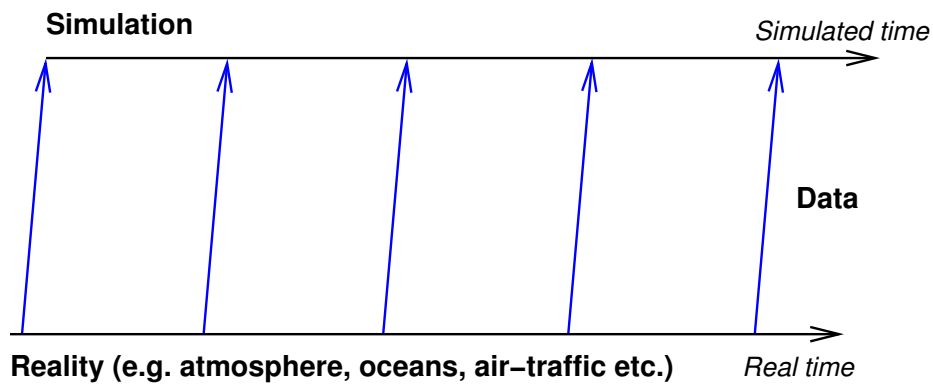


Figure 1: Simplest form of data driven simulation

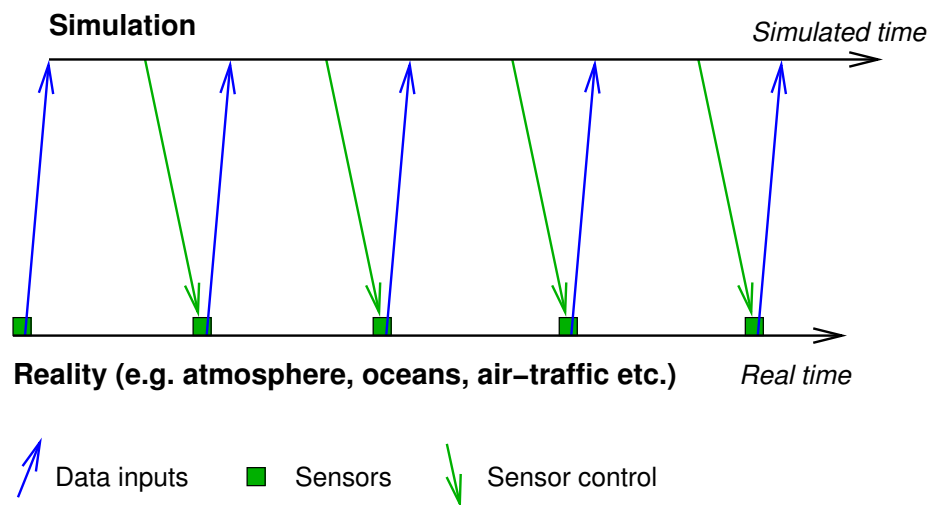


Figure 2: Data driven simulation with data selection

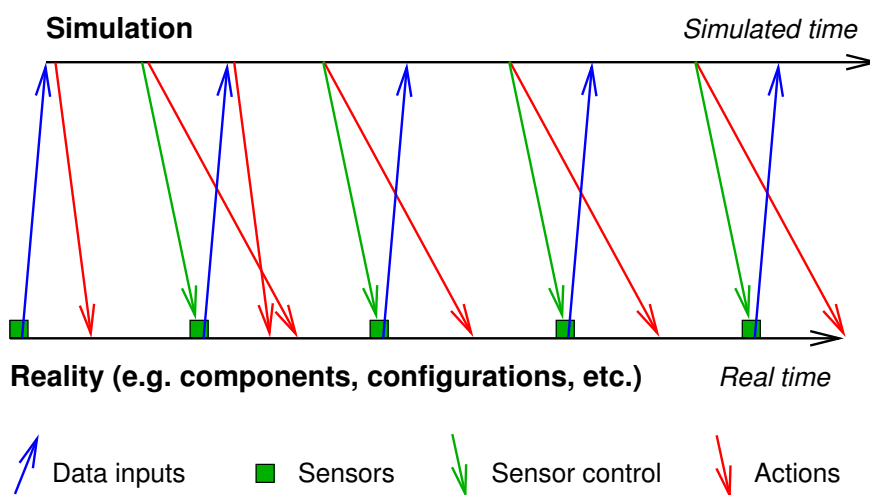


Figure 3: Symbiotic Simulation

This is used by a software agent (labelled Discovery Assistant) to select relevant data and to suggest possible model revisions, possibly by generating its own model (M), which is an evolving (modifiable) version of D. It may also autonomously adapt the simulation, but this is included as optional (dotted

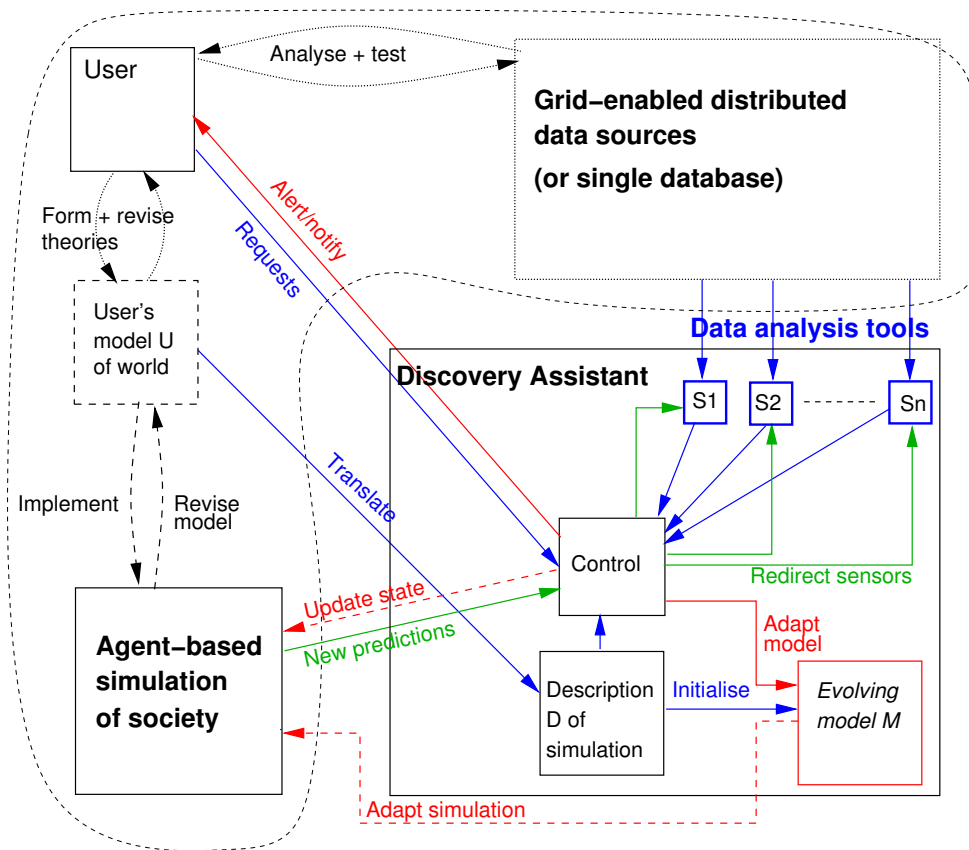


Figure 4: AIMSS schematic architecture

lines). Analogous to the pure DDDAS architecture, there is a symbiotic feedback system between the following processes (labelled S1 and S2 because they are specialised):

- (S1) *Data selection*: predicted states of the social simulation are used to form database queries and to interface with data analysis and mining tools to inquire whether there is evidence for the predicted state. The “predictions” are states that would be expected to exist now if the model’s assumptions are true. If insufficient data is available, the discovery agent can suggest new kinds of data that are required in future surveys.
- (S2) *Adaptation*: results of database queries may be fed back into the simulation. Persistent discrepancies between the simulation predictions and the results of the data analysis are a trigger for the discovery agent to suggest model revisions or to try to autonomously adapt the simulation.

We are currently developing an experimental software prototype, which implements some of the components in Figure 4. There are four different components that can be implemented:

1. case study and agent-based simulation (based on user model **U**);
2. data sources for the case study;
3. data analysis and mining tools;
4. a software agent (Discovery Assistant) with the following capabilities:
 - (a) to interpret the simulation and data using an internal description (**D**) which is based on an ontology for the case study;
 - (b) to select data from given sources or suggest new sources according to aspects of the simulation predictions that are particularly relevant to policy goals (fits into S1 above);

- (c) to compare simulation predictions with observations from data and to detect discrepancies (necessary for S2);
- (d) to adapt the simulation parameters and if necessary the underlying model so that simulation predictions are more closely aligned with observations (necessary for S2).

This report describes progress on implementing components 1-3 as well as 4(a) and 4(c). A subsequent report will cover 4(b) and 4(d).

2 Related Work

LEAD [20] and POSEIDON [19] are examples of existing DDDAS projects with similar goals to the architecture we are developing, except that they are simulating a physical environment and are absorbing real-time data. Details of these projects are in the earlier report [10]. They do not include agent-based simulations but involve the important aspects of online adaptation and data selection. [3] reports on an ongoing DDDAS project which includes agent-based simulations by integrating them with physical simulations (fire evacuation). The main focus of this project is the interaction between the agent-based model and the physical model of a spreading fire within a building. Integration with physical simulations is important in a more advanced application of our architecture, in which human interaction with the environment plays an important role.

In the general domain of scientific assistance (apart from DDDAS), *PolicyGrid* [7] is a rigorous framework for repeatable experimentation with social simulations using Grid services. The Policy-Grid equivalent of D is a description of the experimental process with its parameters, hypotheses and results. In contrast to the AIMSS approach the ontology used for the description is not about the *content* of simulations or data (i.e. no social science domain knowledge), but is instead about the scientific process. We expect that an AIMSS type architecture can be integrated with a PolicyGrid framework.

[12] reviews AI methods to support scientific discovery (e.g. data and text mining) and emphasises the role of human interaction in a semi-automated discovery process. Many of the techniques reviewed there can be integrated into an AIMSS framework.

3 A Housing Case Study

The first component in the architecture is an agent-based simulation based on a user model U . As an example case study, we are modelling the behaviour of agents in the housing domain. In particular, we focus on their reasons for moving between or within neighbourhoods. In this case, the user model U is a set of assumptions about the nature of different households and environments, including assumptions about needs of households and their behaviour in differing circumstances. Such a model can be divided into two parts:

1. an *ontology* defining what entities exist and relations between them: this is an important foundation of the model, although it does not need to be explicitly defined;
2. behaviour of the entities that can be simulated.

For the purposes of this prototype, we use an informal ontology with the following entities:

1. entities that can make decisions and act: these are the agents. Currently these are only households, each represented as a single agent (“household reference person”);
2. possible actions that an agent can take (currently only moving house);
3. needs of households (home availability, affordability, space, safety, health, services): a “need” is a critical requirement that an agent will try to satisfy;

4. household parameters that can change (income, number of persons, age, health, current home condition, etc.);
5. entities in a household's environment (current home, neighbourhood, town/city, etc.);
6. environmental parameters that can change (number of homes, number of vacant homes, house prices, level of crime, service provision, environmental pollution etc.)

Parameters are attributes of an entity. They can either change dynamically (e.g. at random intervals or as a result of simulated physics) or they can change as a result of agent activity. Clearly some entities and their parameters can be further refined (e.g. "services" can be further divided into schools, hospitals, public transport, leisure etc.) The example model is not intended to be prescriptive for the social sciences, but used merely for the purposes of testing.

3.1 A simple agent-based simulation

We defined the agent behaviour in the form of simple "if-then" rules. We deliberately limited the rules so that we know that they are incomplete. The eventual aim is to discover new rules that can be added to the model thus making it a more accurate representation of the reality.

For the first version of the prototype, an agent wants to move only for the following reasons:

1. the current property is not affordable (rent or mortgage too high) or
2. the current property is overcrowded;

The first overrides the second. The agent will move to an overcrowded place if it is the only one that is affordable. However, it will still not be "happy". I.e. its critical requirements are not met. Currently these critical requirements are only affordability and space. We are expanding the simulation to include other criteria such as health, environment services etc. As long as an agent is not happy it will keep on attempting to move.

3.1.1 Agent environment

We have implemented the agent-based simulation in an extremely simplified and abstract environment using RePast². The aim is to represent important features of a typical city in a very generalised way. Currently the agent "space" is divided into 4 quadrants:

- Q1. Expensive city centre apartments, quite small, densely populated;
- Q2. Inner city towerblocks, inexpensive, cramped, densely populated;
- Q3. Modest suburb (medium sized with gardens), moderately populated;
- Q4. Wealthy suburb (large expensive houses with large gardens), sparsely populated.

This is currently being made more general so that different kinds of region and subregion can be specified as parameters.

At initialisation, homes ("properties") are allocated randomly to these quadrants with largest number in inner city and city centre. Precise densities can be specified as parameters.

A household is treated as a single agent although it may have more than one member. Households are allocated randomly to quadrants initially with varying densities. (e.g. Q1: all properties occupied; Q2-Q4: 1 in 5 vacant).

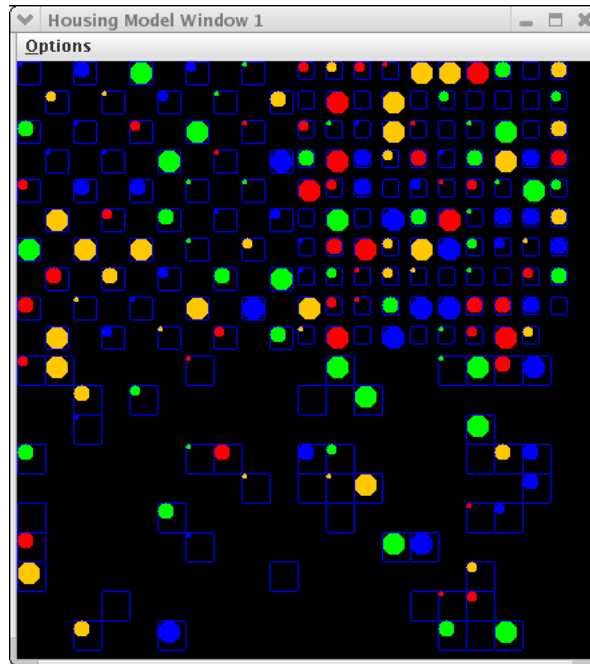


Figure 5: Initial state: 0 steps

3.1.2 An example run

Figures 5, 6 and 7 show a series of snapshots of the simulation. In all figures, a property is shown as a square (small or large) and a household is shown as a circle (also small or large indicating household size from 1-4) inside a property. If a household is not happy and wants to move, it is shown as a filled-in square shape, otherwise it is a filled-in circle (all are happy initially). Household income is colour coded: green = very high, blue = high, orange = medium and low = red.

The simulation is a sequence of steps in which agents decide whether they want to move based on whether they can afford their current home and whether it has enough space (according to the above rules). Figure 5 shows the initial state, where agents are allocated randomly to properties regardless of income and household size. They are initially all in the “happy” state. After the first step of the simulation, many agents decide that they are not happy and want to move. Those that find available properties move into them, but only a limited number are vacant. Therefore many agents remain “unhappy”, even if they have moved, because their needs are still not met. This is shown in Figure 6.

Figure 7 shows the simulation state after a number of steps. Since this version of the simulation is very limited and parameters such as income and household size do not spontaneously change, the simulation states change very little with subsequent steps, except that those households that are low income tend to move within the same area and remain unhappy.

Note, however, that even with such a simple simulation there are some unexpected emergent properties. One of these is the tendency for many empty properties in the “city centre” quadrant. (The accumulation of empty properties on the left of the quadrant is an artifact of the programming and should be ignored. However, the fact that there are many empty properties is a consequence of the properties being small and expensive, resulting in fewer agents wanting to occupy them than for the other quadrants).

²<http://repast.sourceforge.net/>

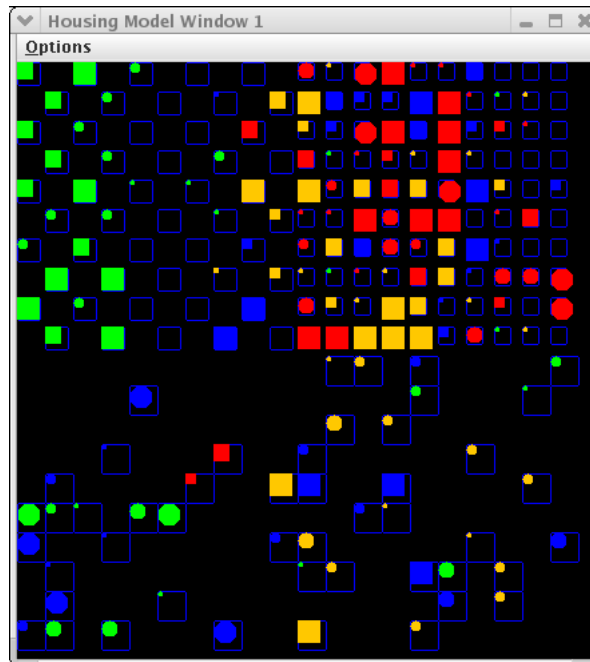


Figure 6: After 1 step

3.2 Data sources

The second component in Figure 4 is a collection of data sources. For our current feasibility study, we are using two databases:

- a database of moves into the social rented sector (tenancies begun with Social Landlords). This is known as the CORE (Continuous Recording) dataset;
- a database of average house prices for each local authority district.

Each CORE record contains the following fields (among others):

- unique identifier, date;
- household details (age, sex, economic status of each person);
- total weekly income of household;
- weekly rent for new tenancy;
- type of property moved into (house, flat etc.), number of rooms;
- location of property moved into (postcode, local authority code);
- previous location (local authority code);
- previous tenure (e.g. social rented, private rented, owner occupier);
- stated reason for move (e.g. affordability, overcrowding, health, poor condition of home, neighbour harassment);

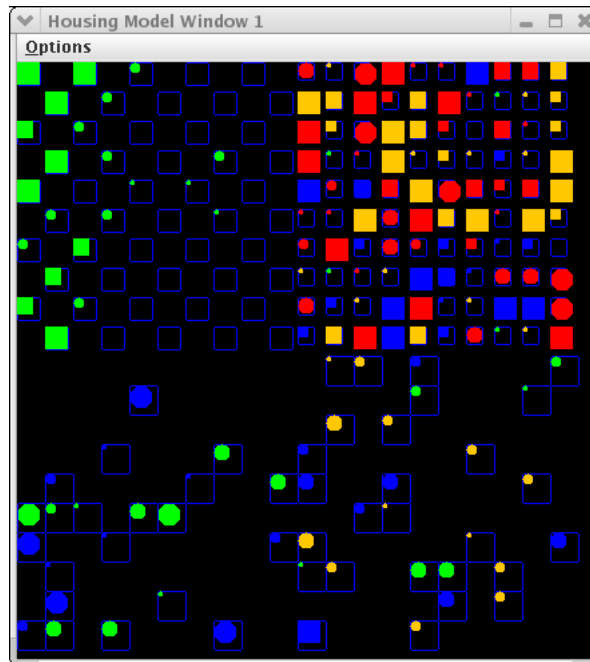


Figure 7: After n steps

4 The AIMSS Discovery Assistant

Since we have an agent-based simulation along with data sources, we can now give a more concrete definition for the discovery assistant (DA) agent that connects them together. For this particular case study, we are modelling a *typical* housing situation on an abstract level, not a particular geographical area such as Birmingham. Therefore the DA will not be making direct comparisons between the simulation predictions and the available data. Instead it will look for *consistency* between the patterns emerging from the simulation and patterns observed in the data. To identify patterns, the analysis tools mentioned in Figure 4 will include data mining tools, as well as data integration and abstraction. In the event of an inconsistency, the appropriate parameters will be adapted in order to bring the simulation into closer alignment with the reality.

4.1 Automated recognition of simulation predictions

After a specified number of steps in the simulation run, the DA agent has to interpret the simulation states and determine its predictions. Since the simulation is abstract, the predictions will be in the form of very general statements that should be true of typical cities. For example: “there are some middle income households in the inner city”, or “there are many vacant properties in the city centre”, where “some” is “at least one” and “many” means significantly more than in other quadrants.

The first version of the simulation “predicted” that moves in the inner city quadrant are very frequent and households are mostly unhappy, while city centre luxury apartments tend to have the most vacancies. Such predictions can be verified or refuted by analysing the data, assuming that the data is typical and from multiple independent sources.

If we run the simulation with no DA agent, such general predictions can only be recognised by humans interacting with the simulation and recognising patterns in the graphics animation (or by looking at charts produced by the simulation). In this case, it would be reasonable to have fully

interactive control of the data mining interface and the DDDAS architecture would be manual.

However, for more complex and realistic simulations (for example in which interaction with neighbourhoods play a key role), human pattern recognition may miss some important emergent properties of the model. The visual results are constrained by the attributes that are selected at design time (e.g. what attributes should be displayed and what should be included in statistical analysis?).

Therefore the automation of component S1 of the DDDAS architecture should as far as possible complement human visual pattern recognition by looking for relations between attributes that may not have been included in the graphics at design time. Of-course it may also be possible to generate new kinds of visualisation of the unexpected patterns.

For this purpose we are currently investigating the application of data mining algorithms to the states of the simulation in order to find interesting predictions that are emergent properties of the model.

4.2 Ontology: Connecting the simulation to the data sources

In order to apply data mining and to reason about its results, the DA agent requires an internal representation of the housing scenario. This is the description D mentioned in Figure 4. The DA agent uses D to interpret the simulation events and to associate attributes and values in the data to the recognised patterns in the simulation. It is important to identify two components:

1. The conceptual framework for the housing model: this is what we call the ontology and defines entities and the relations of the model. For example, households and homes exist and a household can move from one region to another; a household has a set of *needs* that must be satisfied;
2. The behaviour model for both the agents and the environment: these are the *dynamic* aspects of the simulation and include the decision rules for the agent as well as probabilistic rules for dynamic changes in the environment and household status (ageing, having children, changes in income etc.) The way in which these entities are *initialised* should also be stated as part of the model, as this requires domain knowledge (e.g. initial densities of population and houses etc.) For more details models, this becomes increasingly non-trivial, see e.g. [2]).

Currently the behaviour model and most of the ontology are implicit in the object oriented design of the agent-based simulation and are not expressed in any machine readable language (such as OWL). This means that they cannot be accessed easily or modified by the DA agent. Only some aspects of the ontology are defined explicitly as abstract attributes within the Java code and are used to interpret and integrate the raw data as well as to interpret the simulation. However, they cannot be easily modified. Currently attributes for households, properties and moves are defined as follows:

- for households: income level (index of 1 to 4) and number of persons (1,2,3, 4 or above);
- for properties: type (flat, house etc.) number of rooms and cost (which may be rent or mortgage);
- for moves: new situation and previous situation, where a “situation” includes: location code, average house price for location, type of tenure (e.g. owner occupier, social rented, private rented); stated reason for move;

For example, a “situation” can include “neighbourhood details” such as crime level, average condition of houses in neighbourhood, condition of environment (pollution, litter etc.), services (e.g. whether there are good schools or health services in the area). We may also include more qualitative attributes such as “sense of belonging” in neighbourhood (e.g. similar ethnic group, similar income level).

The attributes are used to interpret both the simulation events and the raw data. We are currently extending the simulation to produce a history of moves so that its events corresponds to records in the CORE data. However, the CORE data can only be compared with a subset of the total moves in the

simulation as the CORE data only involves moves into the social rented sector. This subset can be identified in the simulation if we ensure that the previous and new “situations” of a simulated move include the tenure type (of which “social rented” is one).

4.2.1 Preprocessing of raw data

To compare simulation predictions with data, the patterns recognised in the data must be expressed using the same concepts as the patterns found in the simulation states (i.e. using the same abstract attributes above). The raw data sources (i.e CORE and house prices) do not map directly onto the abstract attributes, but must be preprocessed. We divided preprocessing into the following stages:

- *Data integration*: related data from different data sources are linked together. For example, the district location code is available in both the CORE and the house price databases and they share the same meanings (with only a few mismatches). Therefore the average house price for the source and destination of a recorded move can be found.
- *Data abstraction*: the values of raw data attributes are used to determine values for the higher level entities defined in the ontology. For example, the number of persons in a household in a CORE data record has to be inferred by the number of ages above 0.
- *Discretization*: many data mining algorithms do not accept numeric values for attributes; instead they must be labels (called “nominal” values) or strings. To convert a numeric attribute such as “income” into a nominal one, the range of values must be divided into intervals, each with a nominal label. For example, income may be divided into “very low”, “low”, “medium” and “high” depending on which interval the numeric value falls into. Determining how to divide numeric values into intervals can be done using a discretization algorithm.

The preprocessing can be achieved by generating datasets both from the simulation and from the raw data, each containing a list of events (moves) expressed in terms of the abstract attributes and their values. So far, we have produced a dataset from the CORE data (integrated with the house price data) and applied a data mining algorithm to it. Table 1 shows the header of a dataset that was generated from the raw data. The dataset is in ARFF format (Attribute Relation File Format)³ and is divided into a header and a body. The header defines the attributes and their set of values. Some attributes in

```
@relation Moves
```

```
@attribute householdSize {1,2,3,'4 or more'}
@attribute incomeLevel {very-low,low,moderate,high}
@attribute newBeds {1,2,3,'4 or more'}
@attribute newRent {cheap,moderate,expensive,very-expensive}
@attribute newPropType {flat,bedsit,house}
@attribute prevTenure {social-rented,private-rented,owner-occupied}
@attribute prevLocHousePrices {'\*(-inf-122467.5)\'', '\*(122467.5-139881)\'',
'\*(139881-153072)\'', '\*(153072-169295)\'', '\*(169295-186386)\'',
'\*(186386-202238)\'', '\*(202238-216592.5)\'', '\*(216592.5-240798)\'',
'\*(240798-279035.5)\'', '\*(279035.5-inf)\''}
@attribute newLocHousePrices {'\*(-inf-122467.5)\'', '\*(122467.5-139881)\'',
'\*(139881-153072)\'', '\*(153072-169295)\'', '\*(169295-186386)\'',
'\*(186386-202238)\'', '\*(202238-216592.5)\'', '\*(216592.5-240798)\'',
'\*(240798-279035.5)\'', '\*(279035.5-inf)\''}
@attribute moveReason {overcrowding,health,poor-condition,
affordability,neighbour-harassment}
```

Table 1: A dataset header showing attributes with possible values

the original data are not included (e.g date and reference number) as they are not nominal. The set of

³<http://www.cs.waikato.ac.nz/~ml/weka/arff.html>

possible values for each attribute is given in {...}. The first two attributes describe the household that is moving and includes the number of persons (householdSize) and income level. The attributes with prefix “new” describe the new situation into which the household is moving (e.g. rent, property type, number of bedrooms and the average house price for the new region. Attributes with prefix “prev” describe the situation that the household is moving from. In the CORE data, less information is held on the previous situation than on the new situation, but “previous tenure” is useful. One of the most important attributes is the “stated reason for moving”, as this can be compared with the effects of agent behaviour rules in the simulation.

The average house prices for new and previous are divided up into intervals. An algorithm was used to divide them up into equal frequencies of occurrence in the house price dataset, given the desired number of intervals as a parameter (in this case 10). The house price dataset is independent of CORE; it is simply a list of statistics for the different local authority regions. Intervals are determined so that they have the same number of occurrences in the dataset, meaning that they have variable sizes. For example, the lowest interval includes all values up to 122,467 because values in this range occur with the same frequency as values in the other ranges. (The term “-inf” is treated as 0). The frequency-based method overcomes some problems with fixed size intervals because the most frequently occurring house prices are more finely discriminated. However, there are still limitations which we will discuss later.

The body of the dataset is a list of elements, each listing values of attributes in the following order (spread over two lines here):

```
householdSize,newBeds,newRent,newPropType,prevTenure,prevLocHousePrices,
newLocHousePrices,moveReason.
```

These are the attributes listed in the header, and in the same order.

Table 2 shows an excerpt of the main body. Each entry in the list (known as an “instance”) denotes

```
@data
.....

3,low,3,moderate,house,social-rented,'\(-inf-122467.5]\'',
'\(-inf-122467.5]\'',overcrowding
1,very-low,2,moderate,flat,private-rented,'\(-inf-122467.5]\'',
'\(-inf-122467.5]\'',health
1,very-low,1,cheap,bedsit,private-rented,'\(122467.5-139881]\'',
'\(122467.5-139881]\'',poor-condition
1,very-low,1,cheap,flat,private-rented,'\(122467.5-139881]\'',
'\(122467.5-139881]\'',poor-condition
1,very-low,1,cheap,flat,social-rented,'\(153072-169295]\'',
'\(122467.5-139881]\'',neighbour-harassment
2,low,1,cheap,flat,owner-occupied,'\(122467.5-139881]\'',
'\(122467.5-139881]\'',health
2,very-low,2,cheap,flat,social-rented,'\(122467.5-139881]\'',
'\(122467.5-139881]\'',poor-condition
```

Table 2: An excerpt from the body of a pre-processed dataset

a move from a source to a destination, along with the stated reason for moving.

4.2.2 Data mining tools

We have done some initial experimentation with Association Rule Mining using the Apriori algorithm [1], which is available in the WEKA Machine Learning package [24]. Association rules are a set of “if ... then” statements showing frequently occurring associations between combinations of “attribute = value” pairs. Association rule mining was chosen because its results are symbolic and thus have the potential to extend an ontology such as that used in D. Furthermore, it is “unsupervised” in the sense that predefined classes are not given. This allows the discovery of unexpected relationships.

Association rules have the following form:

if (a_1 and a_2 and ... and a_n) then (c_1 and c_2 and .. c_m)

where a_1, \dots, a_n are *antecedents* and c_1, \dots, c_m are *consequents* of the rule. Both antecedents and consequents have the form “attribute = value”. Both the LHS and RHS are conjunctions of “attribute=value” pairs. n and m can be any number up to a specified maximum. The values are mutually exclusive as they are either strings or nominal labels for discrete intervals. Table 3 shows some output

```
Total instances added to list = 2172
Total instances read from database = 8797

Apriori
=====

Minimum support: 0.1 (217 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 18

Generated sets of large itemsets:

Size of set of large itemsets L(1): 28
Size of set of large itemsets L(2): 87
Size of set of large itemsets L(3): 76
Size of set of large itemsets L(4): 31
Size of set of large itemsets L(5): 2

Best rules found:

1 newLocPrices='3' 247 ==> prevLocPrices='3' 238 (0.96)
2 newRent=cheap newLocPrices='2' 297 ==> prevLocPrices='2' 286 (0.96)
3 prevTenure=social-rented newLocPrices='2' 254 ==> prevLocPrices='2' 244 (0.96)
4 newLocPrices='2' 386 ==> prevLocPrices='2' 370 (0.96)
5 newRent=cheap prevLocPrices='2' 299 ==> newLocPrices='2' 286 (0.96)
6 newLocPrices='1' 333 ==> prevLocPrices='1' 318 (0.95)
7 prevLocPrices='2' 390 ==> newLocPrices='2' 370 (0.95)
8 prevTenure=social-rented prevLocPrices='2' 258 ==> newLocPrices='1' 244 (0.95)
9 newRent=cheap newLocPrices='1' 274 ==> prevLocPrices='1' 259 (0.95)
10 newRent=cheap prevLocPrices='1' 274 ==> newLocPrices='1' 259 (0.95)
```

Table 3: Example output from Apriori algorithm

from the pre-processing and analysis of the data for mining. The first two lines show the number of elements inserted into the newly generated dataset (“moves”). This is the result of pre-processing from the raw data (including integration of CORE with house price data). The second line gives the (much larger) number of records that were originally read from the CORE data. Only a fraction of these are used because we specified parameters to select only recognised values for attributes (corresponding to those given in Table 1). For example, only those CORE records were selected where “reason for move” is one of those given. The CORE data includes other values for attributes, including missing values and nonspecific options such as “other” which we filtered out for the purposes of testing.

The second part of Table 3 shows the initial output of the Apriori algorithm when it was applied to the newly generated dataset. An *itemset* is a list of (attribute=value) pairs that can be used to form association rules. The length of the itemset is the number of pairs it contains. The algorithm cycles through the dataset, first collecting all itemsets of length 1, then of length 2 etc. The itemsets of different lengths found in the dataset are shown in Table 3.

Two parameters are specified. *Support* is the number of occurrences of an itemset in the dataset. Only those itemsets with minimum support are considered as candidates for construction of association rules. The *confidence* of a rule is the number of times the right hand side is true, given that the itemset on the left is true. It is effectively the accuracy of the rule in predicting the consequents, given the antecedents. The higher the support and confidence of a rule, the more it represents a regular pattern in the dataset.

The third part of Table 3 shows some example association rules generated by Apriori, when given the dataset in Table 2. The Apriori format of each rule is as follows:

LHS s_1 => RHS s_2 c

where LHS and RHS are left side and right side respectively, s_1 is the support for the LHS, s_2 is the support for the RHS given the LHS and c is the confidence (s_2/s_1). Note that the conjunction between the (attribute=value) pairs is represented by a space. The “if .. then” is represented by an implication sign =>. For space reasons, the house price intervals have been indexed with ‘1’, ‘2’, up to ‘10’ (with the lowest indices representing lowest intervals) and some of the verbose output of Apriori has been reduced.

4.2.3 Ensuring relevance

It is possible to ask specific questions relating to Public Policy. One “discovery” made by Apriori in the above table is that most moves into the social housing sector take place within or into a low house price area. We may want to ask the question: “what kind of new situation do the social housing tenancies lead to, given that the old situation is deprived or disadvantaged?”. After increasing the maximum number of rules to 50 and decreasing the minimum confidence to 0.85, the following rule was obtained:

```
11. prevLocHousePrices='1' 337 ==> newLocHousePrices='1' 318 conf:(0.94)
```

This means that if the house prices in the previous location are in the lowest category then the new one also tends to in the lowest category. Another rule is:

```
29. householdSize=4 moveReason=overcrowding 328 ==>
newPropType=house 287 conf:(0.88)
```

If the household is large and wanted to move due to overcrowding then the new tenancy is a house. This suggests that overcrowding tends to be alleviated by housing association tenancies. However, this may be insufficient evidence as the confidence is only 0.88 and the property may still have too few bedrooms even if it is a house.

Many of the rules shown in Table 3 are not very useful, as they only state what is already obvious. Postprocessing methods are available to remove redundant or irrelevant rules. For example, [23] allows biologists to filter out irrelevant rules by specifying templates. Statistical significance testing may also be used in postprocessing. However, a rule with a low “support” value in a particular database may not be statistically significant but it may point to an important relationship and could be used to focus the search for more data.

5 Ongoing and Future Work

For the software prototype (which implements part of the AIMSS architecture), work is currently ongoing on the following:

- Extension of simulation to include a larger set of needs (health, safety, environment, services) as well as spontaneous dynamic events not modelled as part of agent decision-making. Such events include changes in household (e.g. income status, household size) and changes in environment (e.g. new houses being built or demolished). In a more complex model, some “dynamic” events could be modelled as agent behaviour (with different kinds of agent).

- Prediction verification.

We will describe prediction verification in more detail. Assuming that the CORE data is “typical” if sampled for a minimum time period (e.g. a year), the simulation can also be sampled for a minimum number of cycles beginning after an initialisation period. Once we have applied data mining to the simulation dataset, the resulting association rules provide the kind of general predictions discussed in Section 4.1. An example is as follows: “if previous location is a high house price region and the household income is low then the new location is a low price region”. To test such a prediction we can apply consistency checking to see if there is a rule that was discovered from the simulation that contradicts a rule that was discovered from the data. This would indicate that the available data does not support the model. Similarly, if any rule agrees with the prediction it would indicate supporting evidence from this particular data. Some existing work on postprocessing of association rules includes contradiction checking. Examples include [14] which uses fuzzy matching of rules against a set of expectations. [18] uses an “unexpectedness” definition of a rule, given previous beliefs. These methods may be applied to an AIMSS type architecture, where the “beliefs” are the predictions of a simulation.

Precise consistency checking would treat each collection of rules as two single statements, each with composed of “if...then..” association rules connected by conjunctions. (These would have been postprocessed to remove irrelevant or redundant rules). Efficient algorithms for general consistency checking are available, e.g. [25]. We are currently investigating the application of this algorithm to our work.

5.1 Towards a general software architecture

Our experience developing the prototype has shown the following limitations:

1. A purely abstract simulation cannot easily be adapted online, as there is no direct correspondence between simulation and reality. However, concept discovery is still possible (as explained in Section 5.1.2).
2. Pre-processing of data is too much determined by artificial boundaries generated by the discretization (e.g. the house prices and income levels are divided up using simple numeric techniques which may overlook natural boundaries). This problem can also be addressed by concept discovery and the use of clustering. (For an introduction to clustering, see [24], section 6.6; for a survey of methods, see [9]).
3. Too little data is available in this case study (and most of the available data is not fully exploited). The ontology should contain a list of available data sources (Grid-enabled).

Each of these limitations are being addressed in the development of a more general software architecture with implementable components and interfaces. In particular, the architecture needs to show how to implement the precise mechanisms of data selection and online adaptation (components 4(b) and 4(d) in Section 1.1). Some of these may be incorporated into the demonstration prototype (if time permits). Current work on these components is detailed below.

5.1.1 Automated configuration of data mining parameters

The rules generated from the simulation should contain a useful predictions to be verified and the rules generated from the data have to make statements about the same entities mentioned in the prediction. It is possible to use the content of the prediction to adjust the parameters of the data mining, e.g. by selecting only certain attributes. Furthermore, the mining algorithm can be run several times with different parameters to produce several sets of association rules, each focusing on a particular area of relevance. This is an example of “evaluation-directed selection” outlined in previous work [11].

The consistency checking may still be inconclusive because there is insufficient data to support or refute the prediction. In this case the internal ontology should contain pointers to additional data

sources, and these may be suggested to the user before data access is attempted. This process of search and exploration of known data sources may continue until the ambiguities are removed. This is an example of “uncertainty-directed” selection in [11].

5.1.2 Exploration and concept discovery

If the consistency-checking continues to be inconclusive, it may be possible to discover new concepts and to add these to the internal ontology which is the basis of D. One simple method of concept discovery is to repeat the pre-processing of the raw data but with a higher “sensitivity”. This means that the pre-processing components for integration and abstraction (mentioned in Section 4.2.1) would have to be configured so that they allow all attributes (and all unknown values of known attributes) in the raw data to be included within the data mining process, even if they are not in D or in the simulation parameters. Even if redundant or irrelevant attributes are filtered out from the data mining parameters, this does not affect any unknown attributes that are present in the input data and hence they may also be present in the resulting rules.

The rules resulting from the data mining algorithm can include these unknown attributes. If they are associated with known attributes (in the association rules), they can be added to an evolving ontology, of which D is only the initial version. In Figure 4, the evolving ontology is implicitly included in the model (M). These provisional concepts may be suggested to the user as a model revision.

The new attributes would be included as new concepts with tentative descriptions and relations to other entities, based on the association rules. This is not expected to be very precise initially, however. For example, if the new concept is “disability” the rules in which it appears may say something about the destination home that the disabled person moves to (e.g. supported housing) but this says little about what kind of entity a “disability” is. The developing concept can be made more precise by searching data sources in which it plays a central role. For example, a target database might record the effects of disability on access to services. Association mining may then be applied to the specialised data sources. Algorithms which implement this kind of process are already used in text mining. For example, the package *Leximancer* [13] generates a “profile” for a named entity depending on its correlations with other concepts. Text mining may be a more precise method of developing new concepts than association rule mining, since text often includes contextual details that are not available in a database.

So far we have discussed a kind of concept discovery in which an unknown label is first encountered in the data; then the system must discover what other known concepts are related to it, and how they are related. An alternative is to discover entirely new concepts using conceptual clustering [16, 22] which is an adaptation of general clustering to the special requirements of qualitative data and the need for clusters to relate to existing concepts. This has been applied to text mining [8]. Ensuring that clusters fit with labels that are used in natural language (which are usually the basis of “top-down” ontologies) is a subject of ongoing research. An analysis will be available in a subsequent report.

5.1.3 Possibilities for adaptation and model revision

There is a range of different possibilities for adaptation. Concept discovery is just the most radical. If we return to the ontology mentioned in Section 4.2, the attributes and behaviour model may be adapted if they are both represented in a machine-readable and modifiable form. Possible forms of adaptation include the following:

- Add new attributes or new values to existing attributes as a result of concept discovery (e.g. instead of filtering out all data with unknown attribute values, add the new values); this could involve a change to an OWL specification.
- Add new behaviour rules as a result of discovering new concepts (attributes);

- Modify the order of execution of behaviour rules, so that a different rule will have precedence. For example, currently the rule with the highest precedence is “affordability”, meaning that if this rule matches the current conditions of an agent, it will override all other considerations (i.e. if it cannot afford where it is living it ignores all other considerations (such as overcrowding) and accepts the first available property regardless of its new situation). Conversely a rule with the lowest precedence would only be acted on if conditions of all preceding rules were not matched by the agent’s current situation (e.g. the agent is not satisfied with the level of services, but it can afford where it is living, it has enough space, it is healthy, etc.)
- Modify the initial values of parameters (such as e.g. initial density of homes in a particular kind of region) or the probabilities used to determine the initial values or to determine when they should change.

Representing these entities in machine-modifiable form is an area of future work.

The kind of modifications to the simulation parameters (including behaviour rules) may not be known in advance. If we take the example of behaviour rules, they could be represented as strings of “condition...action” rules, each with a priority attached to it, indicating its place in the order of execution. (Note that “condition ... action...” rules are intended to give a causal explanation of behaviour and not the same as association rules, which merely show correlations).

Populations of such strings of behaviour rules may be subjected to an evolutionary algorithm (such as genetic algorithms [17]) to evolve a simulation that is most consistent with the reality in terms of behaviour. Behaviour models that are most “fit” can be regarded as good explanations of the observed data. However, domain experts would have to interact with the system to filter out unlikely behaviours that still fit the available data.

6 Added Value for Social Scientists

It is important to show that the DDDAS methodology has added value for social scientists and policy-makers. Two features, in particular, are expected to provide advantages:

1. Combination of simulation and data mining to generate and test predictions, and to extend or revise models (including ontologies if necessary);
2. Semi-automation of the processes involved in (1);

In the first case, simulation and data mining along with visualisation could be used to implement a fully manual version of the DDDAS process, which can be understood as “evidence based model development”. This is expected to be an iterative process with the following stages:

1. Formulate initial model and run simulation(s);
2. After a certain time (after the simulation has stabilised) analyse the simulation results using different forms of visualisation and determine what it is predicting. If there is a prediction that is particularly relevant to policy goals, this can be tested.
3. Using a data mining package (along with other data analysis tools) determine which data sources are relevant, and (if necessary) which data attributes within each source should be analysed in order to verify the prediction.
4. Collect and integrate the data using Grid-enabled data access methods.
5. Apply preprocessing to the data (e.g. using discretization and filtering).
6. Run the algorithm(s) on the data.
7. Interpret the results of the data mining and determine whether they are consistent with the prediction.

8. If there is a discrepancy, determine whether the model should be revised. Experiment with variations of the original simulation and return to Step 2.

6.1 Why not just use data mining?

It is possible to only use data mining and machine discovery to generate models. Fully undirected use of data mining can uncover hidden structure and relationships in the data. For example, clustering ([24], section 6.6) may be used in addition to association rule mining. Methods of visualisation are available to produce spatial diagrams of clusters and how they relate. These relationships can be used as predictive models. For example, an association rule can be a predictive rule with the form: “if the left hand side is true then the right hand side will also be true”. In *Predictive Apriori* [21] the confidence of an association rule is combined with support to give a measure of the predictive accuracy of the rule.

However, agent-based simulations can add value to the process of model-generation for the following reasons: *First*, automatically generated models from undirected data mining are developed without the help of domain expert knowledge (i.e. social science knowledge). In other words, they are “bottom-up”. As a result they can make predictions that are irrelevant or contain no new information. By contrast, descriptive agent-based models can make full use of domain expert knowledge, as well as “common sense” [6].

Secondly, running agent-based models as simulations can give rise to possible explanations of *why* some events occur. E.g. “severe housing shortage in a particular area will result *because* the schools in the area are above average standard and more families will want to move there”. This is particularly true of non-numeric, symbolic approaches to agent-based simulations [5].

Thirdly, the consequences of a model’s rules when applied in a particular set of circumstances are often not known in advance (this also applies to numerical modelling, where a “rule” can be a physics equation). Running the model as a simulation can show up some unexpected emergent properties (as detailed in Section 4.1. These properties may then be used as predictions that can lead to a focused use of data mining and analysis tools to ask specific questions. Of-course it is possible to generate predictions without the use of simulation, and to use data mining and other analysis tools in a directed way (e.g. if only certain attributes are selected). However, the specific results from a simulation can lead to unexpected combinations of attributes being selected and may even lead to new ways of combining data sources that would otherwise not have been considered (e.g. because of assumptions of them being unimportant).

6.2 Why automate the model building and testing process?

The added value of automation of the model building process is a topic of ongoing research. We expect the following advantages:

1. Adaptation of a simulation to the data could be done by running many simulations in parallel and applying an evolutionary algorithm to find the one that best matches the data, thus saving considerable time. Even for one simulation, modifying many parameters using a graphical user interface is very time-consuming and error-prone.
2. Interpretation of simulation predictions: unexpected emergent properties of the simulation may not be covered by visualisation methods. An automated interpretation can also be valuable to draw attention to hidden emergent properties (as mentioned in Section 4.1.
3. Detailed comparisons between simulation predictions and results from data analysis may be error prone and time-consuming as is the configuration of data mining parameters.
4. Automated configuration of data integration and mining may deliver surprising results that may not have been discovered otherwise.

These processes are not expected to be fully automated. For example, visualisation can be combined with automated interpretation.

7 Summary and Conclusion

The AIMSS architecture contains four main components: simulation, data sources, data mining tools and a “Discovery Assistant” (DA) agent which interprets simulation predictions and tests them against the data. In our demonstration prototype, we have implemented the first three components (which have been relatively easy) along with a simplified DA agent which connects the simulation predictions with the results of data mining and analysis. Future work will involve the development of mechanisms for online adaptation of the simulation.

Acknowledgements

This work is funded by the Economic and Social Research Council as part of the e-Social-Science Small Grant Scheme.

References

- [1] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules in Large Databases. In *Proceedings of the International Conference on Very Large Databases*, pages 478–499, Santiago, Chile: Morgan Kaufmann, Los Altos, CA, 1994.
- [2] Mark Birkin, Andy Turner, and Belinda Wu. A Synthetic Demographic Model of the UK Population: Methods, Progress and Problems. In *Second International Conference on e-Social Science*, Manchester, UK, June 2006.
- [3] Alok Chaturvedi, Angela Mellema, Sergei Filatyev, and Jay Gore. DDDAS for Fire and Agent Evacuation Modeling of the Rhode Island Nightclub Fire. In *Workshop on Dynamic Data-Driven Applications Simulation at ICCS 2006, LNCS 3993*, pages 433–439, Reading, UK, May 2006. Springer-Verlag.
- [4] Frederica Darema. Grid Computing and Beyond: The Context of Dynamic Data Driven Applications Systems. *Proceedings of the IEEE: Special Issue on Grid Computing*, 93(3):692–697, March 2005.
- [5] Bruce Edmonds. Against the Inappropriate Use of Numerical Representation in Social Simulation. Technical Report CPM-04-129, Centre for Policy Modelling, Manchester Metropolitan University, 2004.
- [6] Bruce Edmonds and Scott Moss. From KISS to KIDS - an Anti-Simplistic Modelling Approach. In *Joint Workshop on Multi-Agent and Multi-Agent-Based Simulation (MAMABS 2004) at the 3rd Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-2004)*, Columbia University, New York City, July 2004.
- [7] P. Edwards, A. Preece, E. Pignotti, G. Polhill, and N. Gott. Lessons Learnt from Deployment of a Social Simulation Tool to the Semantic Grid. In *First International Conference on e-Social Science*, Manchester, UK, June 2005.
- [8] A. Hotho and G. Stumme. Conceptual clustering of text clusters. In *Proceedings FGML Workshop (Fachgruppe Maschinelles Lernen)*, Hannover, 2002.
- [9] A. K. Jain, M. N. Murty, and P. J. Flynn. Data Clustering: A Review. *ACM Computing Surveys*, 31(3), September 1999.
- [10] C. M. Kennedy and G. K. Theodoropoulos. Towards Intelligent Data-Driven Simulation for Policy Decision Support in the Social Sciences, 2005. Technical Report CSR-05-9, School of Computer Science, University of Birmingham.

- [11] Catriona Kennedy and Georgios Theodoropoulos. Intelligent Management of Data Driven Simulations to Support Model Building in the Social Sciences. In *Workshop on Dynamic Data-Driven Applications Simulation at ICCS 2006, LNCS 3993*, pages 562–569, Reading, UK, May 2006. Springer-Verlag.
- [12] Pat Langley. The Computational Support of Scientific Discovery. *International Journal of Human-Computer Studies*, 53(3):393–410, 2000.
- [13] Leximancer Concept Miner: From Words to Meaning. White paper at: http://www.leximancer.com/documents/leximancer_white_paper.html, Last visited October 2006.
- [14] Bing Liu and Wynne Hsu. Post-analysis of learned rules. In *AAAI/IAAI, Vol. 1*, pages 828–834, 1996.
- [15] Malcolm Yoke Hean Low, Kong Wei Lye, Peter Lendermann, Stephen John Turner, Reman Tat Wee Chim, and Surya Hadisaputra Leo. An Agent-based Approach for Managing Symbiotic Simulation of Semiconductor Assembly and Test Operation. In *Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS 2005)*, Utrecht, The Netherlands, July 2005.
- [16] R. S. Michalski and R. E. Stepp. Learning from Observation: Conceptual Clustering. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning: An artificial intelligence approach*, pages 331–363. Morgan Kauffmann, Palo Alto, CA:Tioga, 1983.
- [17] Melanie Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, 1998.
- [18] Balaji Padmanabhan and Alexander Tuzhilin. A Belief-Driven Method for Discovering Unexpected Patterns. In *Knowledge Discovery and Data Mining*, pages 94–100, 1998.
- [19] N.M. Patrikalakis, J.J. McCarthy, A.R. Robinson, H. Schmidt, C. Evangelinos, P.J. Haley, S. Lalis, P.F.J. Lermustaux, R. Tian, W.G. Leslie, and W. Cho. Towards a dynamic data driven system for rapid adaptive interdisciplinary ocean forecasting. In F. Darema, editor, *Dynamic Data-Driven Application Systems*. Kluwer Academic Publishers, Amsterdam, 2004.
- [20] Beth Plale, Dennis Gannon, Dan Reed, Sara Graves, Kelvin Droegemeier, Bob Wilhelmson, and Mohan Ramamurthy. Towards Dynamically Adaptive Weather Analysis and Forecasting in LEAD. In *Workshop on Dynamic Data Driven Application Systems at the International Conference on Computational Science (ICCS 2005)*, Atlanta, USA, May 2005.
- [21] Tobias Scheffer. Finding Association Rules that Trade Support Optimally against Confidence. *Lecture Notes in Computer Science*, 2168:424+, 2001. Proceedings of the European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD-01). 2001.
- [22] R. E. Stepp and R. S. Michalski. Conceptual Clustering: Inventing Goal Oriented Classifications of Structured Objects. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning: An artificial intelligence approach, volume II*. Morgan Kauffmann, San Mateo, CA, 1986.
- [23] Alexander Tuzhilin and Gediminas Adomavicius. Handling very large numbers of association rules in the analysis of microarray data. In *Eighth International Conference on Knowledge Discovery and Data Mining (KDD2002)*, 2002.
- [24] Ian H. Witten and Eibe Frak. *Data Mining: Practical Machine Learning Tools and Techniques*. Elsevier, San Fransisco, California, 2005.
- [25] zChaff. Boolean Satisfiability Research Group at Princeton <http://www.princeton.edu/~chaff/zchaff.html>. Last consulted October 2006.